

Error Floor Approximation for LDPC Codes in the AWGN Channel

Brian K. Butler, *Senior Member, IEEE*, Paul H. Siegel, *Fellow, IEEE*

Abstract—This paper addresses the prediction of error floors of variable-regular Low Density Parity Check (LDPC) codes in the Additive White Gaussian Noise (AWGN) channel. Specifically, we focus on the Sum-Product Algorithm (SPA) decoder in the log-domain at high SNRs. We hypothesize that several published error floor levels are due to numerical saturation within their decoders when handling high SNRs. We take care to develop a log-domain SPA decoder that does not saturate near-certain messages and find the error rates of our decoder to be lower by at least several orders of magnitude. We study the behavior of near-codewords / trapping sets that dominate the reported error floors.

J. Sun, in his Ph.D. thesis, used a linear system model to show that error floors due to elementary trapping sets don't exist under certain conditions, assuming that the SPA decoder is non-saturating [1]. We develop a refined linear model which we find to be capable of predicting the error floors caused by elementary trapping sets for saturating decoders. Performance results of several codes at several levels of decoder saturation are presented.

Index Terms—Low-density parity-check (LDPC) codes, belief propagation (BP), error floor, linear analysis, Margulis code, absorbing sets, near-codewords, trapping sets.

I. INTRODUCTION

A very important class of modern codes, the Low Density Parity Check (LDPC) codes, was first published by R. Gallager in 1962 [2], [3]. LDPC codes are linear block codes described by a sparse parity check matrix. Decoding algorithms for LDPC codes are generally iterative. The renaissance of interest in these codes began with work by D. MacKay, R. Neal, and N. Wiburg in the late 1990s [4], [5]. Progress has been rapid, with theoretic channel capacity essentially reached in some examples and standardization complete for commercial use in others, *e.g.*, DVB-S2 for satellite broadcast and IEEE 802.3an for 10 Gbit/s Ethernet.

An iteratively decoded code's error rate performance versus channel quality graph is typically divided into two regions. The first region, termed the *waterfall*, occurs at poorer channel quality, close to the decoding threshold, and is characterized by a rapid drop in error rate as channel quality improves. The second region, called the *error floor*, is the focus of this paper. The error floor appears at higher channel quality and is characterized by a more gradual decrease in error rate as

channel quality improves. For non-ML, iterative decoders the error floor is apparently determined by small structures within the code that are specific to the selected graphical description of the code.

The understanding of the LDPC error floor has progressed significantly, but issues remain. For the binary erasure channel (BEC), the structures that limit the iterative decoder's performance as channel conditions improve are known as *stopping sets* [6]. These sets have a combinatorial description and can be enumerated to accurately predict the error floor for the BEC.

Other memoryless channels are more difficult to characterize and the presence of error floors has limited the adoption of LDPC codes in some applications. For the DVB-S2 standard, an outer algebraic code is concatenated with LDPC to ensure a low error floor. MacKay and Postol found that *near-codewords* caused the error floors of the Margulis-type codes in the AWGN channel in their belief propagation decoder [7]. Richardson wrote the seminal paper on the error floors of memoryless channels shortly afterwards [8]. In it, he called near-codewords *trapping sets* and defined them with respect to the decoder in use as the error-causing structures. The parameters (a, b) are used to characterize both near-codewords and trapping sets, where a is the number of variable nodes in the set and b is the number of unsatisfied check nodes when just those variable nodes are in error. The (a, b) parameters of error-floor-causing structures are typically small.

Richardson emphasized techniques for the AWGN channel in [8]. He detailed a methodology to predict a trapping set's impact by simulating the AWGN channel in the neighborhood of the set. His semi-analytical technique was shown to be accurate at predicting the error floors given that the trapping sets were known. The method involved significant simulation, but much less than standard Monte Carlo simulation. Roughly speaking, error floors can be measured down to frame error rates of about 10^{-8} in standard computer simulation and about 10^{-10} in hardware simulation, depending on code complexity and resources available. Richardson's method allows us to reach orders of magnitude further in characterizing the error floor.

Further work on trapping sets includes algorithms to enumerate the occurrence of small trapping sets in specific codes [9]–[11] and the characterization of trapping set parameters in random ensembles of LDPC codes [12].

Several other significant works on error floors in the AWGN channel exist. Dolecek et al. noted empirically that the trapping sets dominating the error floor had a combinatorial description termed *absorbing sets* [13]. That work was on hardware-oriented Sum-Product Algorithm (SPA) decoders. Also, a

This work was presented in part at the Forty-Ninth Annual Allerton Conference, Allerton House, Illinois, Sept 2011.

The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego (UCSD), La Jolla, CA 92093 USA (e-mail: butler@ieee.org, psiegel@ucsd.edu).

This work was supported in part by the Center for Magnetic Recording Research at UCSD and by the National Science Foundation (NSF) under Grant CCF-0829865 and Grant CCF-1116739.

variety of works exist on techniques to overcome error floors. In this area, we note [14] proposes several such techniques, [15] slows decoder convergence using averaged decoding, [16] selectively biases the messages from check nodes, [17] employs informed scheduling of nodes for updating, and [18] adds check equations to the parity check matrix. This paper makes no changes to the standard SPA decoding aside from fixing common numerical problems.

There has been work on analytical techniques to predict error floors in AWGN. Sun developed a model of elementary trapping set behavior based on a linear state-space model of the trapping set with density evolution applied to the code outside of the trapping set [1], [19]. This work has gained little attention, even though it reaches a conclusion contrary to prior results. Sun's models show no error floor for elementary trapping sets (excluding codewords) in regular, infinite-length LDPC codes if the variable-degree of the code is at least three and the decoder's metrics are allowed to grow very large. Sun is able to make similar claims for irregular LDPC codes under certain conditions. In these cases, Sun shows that the graph outside of the trapping set will eventually correct the trapping set errors; in his example, using a (3,1) trapping set, this occurred at a mean log-likelihood ratio (LLR) of about 10^{10} after about 40 iterations. In his Margulis code example, Sun estimates a nonzero, but very low error floor for a non-saturating decoder using his approximation to SPA. He attributes this error floor to the code's finite length.

Sun's analytical development was asymptotic in the number of iterations. We work to identify the conditions when the growth of the check nodes' beneficial LLRs from outside of the trapping set can out-pace the internal growth of the trapping set's detrimental LLRs. In doing so, we develop new signal-to-noise ratio (SNR) thresholds that indicate when this LLR growth regime is reached. We also show how the independence assumption breaks down when applied to finite-length codes with cycles due to strongly correlated LLR messages when the decoder is non-saturating.

Schlegel and Zhang [20] extended Sun's work by adding time-variant gains and channel errors from outside of the trapping set to the model. They then matched predicted error floors to error floors observed in hardware simulation, specifically for the code in 802.3an. Their results show that the external errors have very little impact and that the error floor can be predicted quite accurately. The model derived in [20] was specific to the dominant (8,8) trapping set of that code. We generalize their model and match predicted error floors to simulations for three different codes with several LLR limits. We find the model estimates the error floor to within 0.2 dB for the three codes we examine. We also show how a model that breaks down for finite length codes may work fairly well when saturated LLRs are introduced to the finite length codes.

Brevik and O'Sullivan developed a new mathematical description of the decoder convergence on elementary trapping sets using a model of the probability-domain SPA [21]. While considering just the channel inputs, they are able to find regions of convergence to the correct codeword, regions of convergence to the trapping set, and regions of non-convergence typified by an oscillatory behavior.

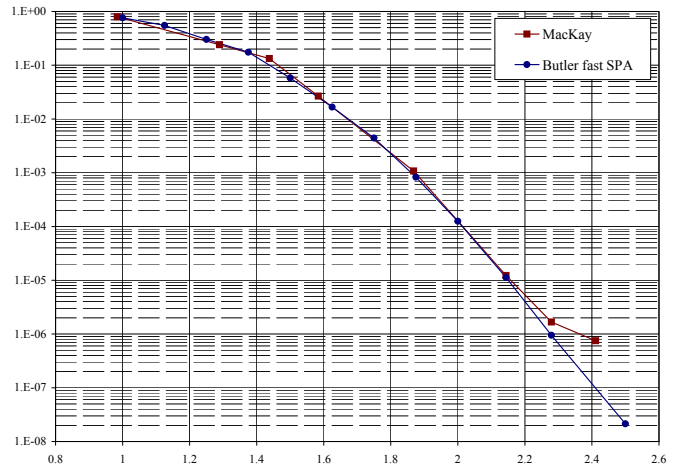


Fig. 1. FER vs. E_b/N_0 in dB for Margulis LDPC Code in AWGN.

Several authors have found empirically that increasing LLR limits in the SPA lowers the error floor level [22]–[26]. We concur, but we believe the effect to be orders of magnitude stronger than noted. We have found that when care is taken to implement the SPA decoder without saturation (or “clipping”), especially at high SNRs, the effect of error floors reduces very dramatically. In the cases we examine, the error floors are reduced by many orders of magnitude. We mostly present the measured error rates in terms of frame error rate (FER) which is equivalent to the codeword error rate in this context. As an example, the FER results of the (2640, 1320) Margulis code from our simulation and [7] are shown in Fig. 1. The results of [7] show a floor starting at about 10^{-6} at an E_b/N_0 of 2.4 dB dominated by near-codewords. Others report even higher error floor levels for the same code, crossing 10^{-6} at about 2.8 dB [8], [14]. As shown in Fig. 1, our simulation has not shown any error floor down to the measurement limit of 2×10^{-8} . The final point at 2×10^{-8} FER amounts to 70 error events in 3060 hours of floating-point simulation, run for a maximum of 200 iterations. None of these error events were of the near-codewords variety that dominated the earlier works: (12,4) or (14,4). This suggests that any error floor in our simulator is substantially lower still. We did see three error events fail on an (18,8) near-codeword. We develop the framework to explain the error floor behavior.

Our focus in this paper will be on the development of error floor predictions over the AWGN channel for binary LDPC codes of regular variable-degree. This is assumed in deriving the results that follow and is an obvious limitation of this work. Section II introduces the general terminology related to nonnegative matrices, graphs, trapping sets, and SPA decoders. We introduce message saturation as a technique to avoid numeric overflow in the SPA and introduce the non-saturating decoder we choose to employ. Section III develops the state-space model we will be using for the analytical results of this paper. Section IV develops the needed properties of the dominant eigenvalue required to characterize the system and the probability model. Section V demonstrates the model's accuracy at predicting saturated performance compared with simulation results for three specific codes. Section VI develops

bounds on unsatisfied-check LLR growth to determine ultimate error floor bounds for non-saturating decoders. Section VII empirically shows the problems of extending the results of the prior section to a finite length code and instead applies a modified version of Richardson's semi-analytical technique to estimate the error floor of a non-saturating SPA decoder. Finally, we draw our conclusions.

II. PRELIMINARIES

A. Nonnegative Matrices

The reader is assumed to be familiar with the basics of linear algebra. In this subsection we introduce the terminology of Perron-Frobenius theory of nonnegative matrices [27]–[29].

A vector $\mathbf{v} \in \mathbb{R}^n$ is said to be *positive* if every element is strictly greater than zero. Likewise, the vector \mathbf{v} is said to be *nonnegative* if every element is greater than or equal to zero. In this paper, we use column vectors by default.

A matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ is said to be a *positive matrix* if every entry is strictly greater than zero. Likewise, the matrix \mathbf{M} is said to be a *nonnegative matrix* if every entry is greater than or equal to zero. These conditions are denoted $\mathbf{M} > \mathbf{0}$ and $\mathbf{M} \geq \mathbf{0}$, respectively. Moreover, the nonnegative matrix \mathbf{M} is said to be a $(0,1)$ -matrix if every entry belongs to the set $\{0,1\}$.

A *permutation matrix* \mathbf{P} is a square $(0,1)$ -matrix, in which each row and each column contain a single 1 entry. A *symmetric permutation* of the square matrix \mathbf{M} is the operation \mathbf{PMP}^T .

The nonnegative square matrix \mathbf{M} is said to be a *reducible matrix* if there exists a permutation matrix \mathbf{P} such that

$$\mathbf{M}' = \mathbf{PMP}^T = \begin{pmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{0} & \mathbf{Z} \end{pmatrix}, \quad (1)$$

where \mathbf{X} and \mathbf{Z} are square submatrices. Otherwise, the matrix \mathbf{M} is said to be an *irreducible matrix*.

Given $\mathbf{M} \in \mathbb{R}^{m \times m}$, the set of distinct eigenvalues is called the *spectrum* of \mathbf{M} , denoted $\sigma(\mathbf{M}) = \{\mu_1, \dots, \mu_n\}$, where $\mu_k \in \mathbb{C}$. The *spectral radius* of \mathbf{M} is the real number $\rho(\mathbf{M}) = \max\{|\mu| : \mu \in \sigma(\mathbf{M})\}$. The *spectral circle* is a circle on the complex-plane about the origin with radius $\rho(\mathbf{M})$.

A nonnegative irreducible matrix \mathbf{M} having only one eigenvalue on the spectral circle is said to be *primitive*. A nonnegative irreducible matrix \mathbf{M} with $h > 1$ eigenvalues on the spectral circle is said to be *imprimitive* or a cyclic matrix. The value h is known as the *index of imprimitivity* or period.

We now summarize a few key results from the theory of nonnegative matrices [27]–[30]. Let \mathbf{M} be an irreducible nonnegative matrix. There is a simple eigenvalue r on the spectral circle which is real, such that $r = \rho(\mathbf{M})$ and $r > 0$. There exists a positive vector \mathbf{v}_1 such that $\mathbf{M}\mathbf{v}_1 = r\mathbf{v}_1$, which is unique up to a positive scale factor. Similarly, there exists a positive vector \mathbf{w}_1 such that $\mathbf{w}_1^T \mathbf{M} = r\mathbf{w}_1^T$, which is unique up to a positive scale factor. When scaled such that $\|\mathbf{v}_1\|_1 = 1$ and $\|\mathbf{w}_1\|_1 = 1$, these vectors are known as the *Perron vector* of \mathbf{M} and the *left Perron vector* of \mathbf{M} , respectively. There are no other nonnegative eigenvectors of \mathbf{M} , except for positive multiples of \mathbf{v}_1 and \mathbf{w}_1 . If \mathbf{M} is imprimitive, the h distinct

eigenvalues on the spectral circle are the h th roots of unity scaled by $\rho(\mathbf{M})$.

Frobenius's test for the primitivity of nonnegative matrix \mathbf{M} is that \mathbf{M} is primitive if and only if $\mathbf{M}^p > \mathbf{0}$ for some positive integer p . There are two tests for the irreducibility of an $m \times m$ nonnegative matrix \mathbf{M} . For each index (i, j) pair, there exists a positive integer $p = p(i, j)$ such that $[\mathbf{M}^p]_{ij} > 0$ if and only if \mathbf{M} is irreducible. The second test is that \mathbf{M} is irreducible if and only if $(\mathbf{I} + \mathbf{M})^{(m-1)} > \mathbf{0}$.

B. General Graph Theory

An *undirected graph* $G = (V, F)$ consists of a finite set of vertices V and a finite collection of edges F . Each edge is an unordered pair of vertices (v_i, v_j) such that the edge joins vertices v_i and v_j . Given the edge (v_i, v_j) , we say that vertex v_i is *adjacent* to vertex v_j , and *vice versa*. A vertex and edge are *incident* with one another if the vertex is contained in the edge.

A loopless graph or *multigraph* does not have self-loops. A *self-loop* is an edge joining a vertex to itself, such as (v_i, v_i) . A *simple graph* has neither self-loops nor parallel edges. *Parallel edges* are multiple inclusions of an edge in the edge collection.

The *order* of a graph is the number of vertices and the *size* is the number of edges. The degree of a vertex $d(v_i)$ is the number of edges incident on the vertex. Euler's handshaking lemma states that the sum of all degrees of a graph must be twice its number of edges. It follows from the fact that each edge in the graph must contribute 2 to the sum of all degrees. Thus, in equation form for the undirected graph $G = (V, F)$, it may be stated that

$$\begin{aligned} \text{order}(G) &= |V| \quad \text{and} \\ \text{size}(G) &= |F| = \frac{1}{2} \sum_{v_i \in V} d(v_i). \end{aligned}$$

A *regular graph* is a graph whose vertices are all of equal degree.

In an undirected graph G , a *walk* between two vertices is an alternating sequence of incident vertices and edges. The vertices and edges in a walk need not be distinct. The number of edges in a walk is its *length*. The vertices v_i and v_j are said to be *connected* if the graph contains a walk of any length from v_i to v_j , noting that every vertex is considered connected to itself. A graph is said to be *connected* if every pair of vertices is connected, otherwise the graph is said to be *disconnected*. The vertices of a disconnected graph may be partitioned into connected components.

Unique edges are those edges which appear once in the edge collection of the graph. A walk that *backtracks* is a walk in which a unique edge appears twice or more in-a-row in the walk.

A *closed walk* is a walk that begins and ends on the same vertex. A *cycle* is a closed walk with no repeated edges or vertices (except the initial and final vertex) of length at least two. The *girth* of a graph is the length of its shortest cycle, if it has cycles. If every vertex in a graph is at least degree two, then the graph contains one or more cycles.

A *leaf* is a vertex of degree one. A *tree* is a connected graph without cycles. Trees, with at least two vertices, have leaves. We call an undirected graph *leafless* if it does not contain leaves.

We will say that two graphs are *identical* if they have equal vertex sets and equal edge collections. Two graphs are said to be *isomorphic* if there exists between their vertex sets a one-to-one correspondence having the property that whenever two vertices are adjacent in one graph, the corresponding two vertices are adjacent in the other graph. This correspondence relationship is called an *isomorphism*. The isomorphism is a relabeling of the vertices that preserves the graph's structure.

The *adjacency matrix* $\mathbf{A}(G)$ of any simple graph G of order n is the $n \times n$ symmetric $(0, 1)$ -matrix whose (i, j) entry is 1 if and only if (v_i, v_j) is an edge of G . More generally, the adjacency matrix $\mathbf{A}(G')$ of any multigraph G' is the symmetric nonnegative matrix whose (i, j) entry indicates the number of edges joining v_i and v_j in G' . The number of walks of exactly length p between vertices v_i and v_j , in graph G , is the (i, j) entry of $\mathbf{A}(G)^p$. Two graphs are isomorphic if and only if there exists a symmetric permutation that relates their adjacency matrices.

A *cycle graph*, denoted C_n , is a multigraph of order $n \geq 2$, in which the distinct vertices of the set $\{v_1, v_2, \dots, v_n\}$ are joined by the edges from the collection $\{(v_1, v_2), \dots, (v_{n-1}, v_n), (v_n, v_1)\}$. A cycle graph is a single cycle.

Lemma 1. *Let G be a connected multigraph of order $n \geq 2$. Then G is isomorphic to the cycle graph C_n if and only if every vertex is degree two.*

Proof: By definition a cycle graph has vertices of degree two. Since isomorphisms preserve structure, graphs isomorphic to a cycle graph also have vertices of degree two. We have proved that graphs isomorphic to a cycle graph are a sufficient condition for every vertex to be degree two. To prove necessity we will take a walk on G , having every vertex of degree two. The walk will not allow backtracks, so at every intermediate vertex there is only one edge to choose to continue walking since every vertex has exactly two edges. The walk will start on vertex v_1 , and end when we return to v_1 . Since the graph is connected this non-backtracking walk must visit every vertex. There are exactly two walks which satisfy these conditions corresponding to the choice of the starting edge selected leaving the first vertex v_1 . If we relabel the vertices we visit in order as we walk v_2, v_3 , etc., our relabeled multigraph is clearly a cycle graph, proving that G is isomorphic to the cycle graph C_n . ■

A *bipartite graph* $B = (V, C, E)$ is a special case of an undirected graph in which the graph's vertices may be partitioned into two disjoint sets V and C . Each edge $e \in E$ of a bipartite graph joins a vertex from V to a vertex from C . Bipartite graphs cannot have self-loops by definition. The parity check matrix \mathbf{H} over $\text{GF}(2)$ describes a bipartite graph B , called the *Tanner graph* of \mathbf{H} , in which the vertices are known as variable nodes V and check nodes C . Tanner graphs of binary codes do not have parallel edges.

A d_v -*variable-regular* Tanner graph is a Tanner graph whose variable nodes all have equal degree d_v , and a (d_v, d_c)

regular Tanner graph is both d_v -variable-regular and has check nodes all of degree d_c . The parity check matrix \mathbf{H} is related to the adjacency matrix of the Tanner graph $\mathbf{A}(B)$ as

$$\mathbf{A}(B) = \begin{pmatrix} \mathbf{0} & \mathbf{H}^T \\ \mathbf{H} & \mathbf{0} \end{pmatrix}. \quad (2)$$

Given a subset of variable nodes $\mathcal{S} \subset V$, we use $\mathcal{N}(\mathcal{S})$ to denote the set of adjacent check nodes. Given a Tanner graph $B = (V, C, E)$ and any set of variable nodes $\mathcal{S} \subset V$, let $B_{\mathcal{S}}$ represents the *induced subgraph* of \mathcal{S} . That is, $B_{\mathcal{S}} = (\mathcal{S}, \mathcal{N}(\mathcal{S}), E_{\mathcal{S}})$ is a bipartite graph containing the variable nodes \mathcal{S} , the check nodes $\mathcal{N}(\mathcal{S})$, and the edges between them $E_{\mathcal{S}}$. We will frequently refer to these induced subgraphs as Tanner subgraphs with parity check submatrix $\mathbf{H}_{\mathcal{S}}$. The submatrix $\mathbf{H}_{\mathcal{S}}$ is formed by selecting the columns of \mathbf{H} as indexed by the members of the set \mathcal{S} and optionally removing any resulting all-zero rows.

A *directed graph* or *digraph* $D = (Z, A)$ consists of a set of vertices Z and a collection of directed edges or *arcs* A . Each arc is an ordered pair of vertices (z_i, z_j) such that the arc is directed from vertex z_i to vertex z_j . For arc (z_i, z_j) , we call the first vertex z_i its *tail* and the second vertex z_j its *head*. We will focus on *simple digraphs*, which exclude self-loops and parallel arcs. Parallel arcs (also called multiarcs) are multiple inclusions of an arc in the arc collection. A directed multigraph or *multidigraph* may contain parallel arcs, but not self-loops.

In a digraph D , a *directed walk* is an alternating sequence of vertices and arcs from z_i to z_j in D such that every arc a_i in the sequence is preceded by its initial vertex and is followed by its terminal vertex. A digraph D is said to be *strongly connected* if for any ordered pair of distinct vertices (z_i, z_j) there is a directed walk in D from z_i to z_j . For example, Fig. 4d is strongly connected.

The *adjacency matrix* $\mathbf{A}(D)$ of any simple digraph D is the $(0, 1)$ -matrix whose (i, j) entry is 1 if and only if (z_i, z_j) is an arc of D .

Lemma 2. *A $(0, 1)$ -matrix is irreducible if and only if the associated digraph is strongly connected.*

Proof: See p. 78 of [29]. ■

The *outdegree* d_i^+ of vertex z_i is the number of arcs in digraph D with initial vertex z_i . Likewise, the *indegree* d_i^- of vertex z_i is the number of arcs with terminal vertex z_i . For the digraph $D = (Z, A)$, we note that

$$\begin{aligned} \text{order}(D) &= |Z| \quad \text{and} \\ \text{size}(D) &= |A| = \sum_{z_i \in Z} d_i^+ = \sum_{z_i \in Z} d_i^-. \end{aligned}$$

A *regular* digraph is a digraph whose vertices are all of equal indegree and outdegree.

The *line graph* $\mathcal{L}(G)$ of multigraph G is the graph whose vertices are the edges of G . Two vertices of $\mathcal{L}(G)$ are adjacent if and only if their corresponding edges in G have a vertex (or two) in common. The *line digraph* $\mathcal{L}(D)$ of multidigraph $D = (Z, A)$ is the digraph whose vertices are the arcs A of D [31]. The arc (a_i, a_j) is in $\mathcal{L}(D)$ if and only if the head of a_i is the tail of a_j in D . The line digraph of any multidigraph

is a simple digraph, and if D is regular then the line digraph $\mathcal{L}(D)$ is regular. For the line digraph $\mathcal{L}(D)$ of multidigraph $D = (Z, A)$,

$$\begin{aligned} \text{order}(\mathcal{L}(D)) &= |A| \quad \text{and} \\ \text{size}(\mathcal{L}(D)) &= \sum_{z_i \in Z} d_i^+ d_i^-. \end{aligned}$$

The interested reader is referred to a more complete treatment of the subject [29], [32], [33]. Our use of graph theory has parallels to [21], from which we borrowed the term “backtrack.”

C. AWGN Channel & SPA

LDPC codes are defined by the null space of a parity check matrix \mathbf{H} . The codewords are the set of column vectors \mathbf{c} , such that $\mathbf{c} \in \mathcal{C}$ satisfies $\mathbf{H}\mathbf{c} = \mathbf{0}$ over a particular field. A given code can be described by many different \mathbf{H} matrices.

The \mathbf{H} matrix over GF(2) may be associated with a bipartite graph $B = (V, C, E)$ termed a Tanner graph described in the previous subsection. The set of variable nodes V represent the symbols of the codeword that are sent over the channel and correspond to the columns of the parity check matrix. The set of check nodes C enforce the parity check equations represented by the rows of \mathbf{H} .

Assumption 1. We are only concerned with codes over the binary field GF(2). Thus, the elements of \mathbf{H} and \mathbf{c} are binary. We consider only codes described by d_v -variable-regular Tanner graphs with $d_v \geq 3$. Also, we assume binary antipodal signaling over the AWGN channel.

After encoding, each binary symbol $c_i \in \{0, 1\}$ is transmitted over the AWGN channel as a binary antipodal symbol $t_i \in \{+1, -1\}$. Every received symbol r_i is simply the sum $t_i + n_i$ of the transmitted symbol plus independent and identically-distributed (i.i.d.) Gaussian noise of zero-mean and variance σ^2 . This yields a channel SNR of $1/\sigma^2$. Equating this measure to traditional bandpass communication figures of merit for coherent detection of binary phase-shift keying we have $1/\sigma^2 = 2E_s/N_0 = 2RE_b/N_0$, where E_s is the received energy per symbol-time, E_b is the received energy per information bit, R is the rate of the code in information bits per binary symbol, and N_0 is the one-sided power spectral density of the noise.

Next, we describe the most broadly used decoding process for LDPC codes with soft-information, the Sum-Product Algorithm (SPA). The SPA decoder would be optimal, in the symbol-wise maximum *a posteriori* (MAP) sense, if the Tanner graph had no cycles. In any “good” finite-length code, the Tanner graph will indeed have cycles, but we generally find that the SPA does quite well. Like many, we prefer to implement our decoder simulation in the log-domain, as it avoids normalization and is closer to the approximations used in building hardware.

We use the notation $\mathcal{N}(i)$ to indicate the set of check nodes adjacent to variable node i . Likewise, we use $\mathcal{N}(j) \setminus i$ to indicate all variable nodes adjacent to check node j , excluding variable node i .

Before starting the decoding process, the received symbols r_i must be scaled to log-likelihood ratios (LLRs) $\lambda^{[i]}$, as

$$\lambda^{[i]} = \ln \frac{P(r_i | t_i = +1)}{P(r_i | t_i = -1)} = \frac{2}{\sigma^2} r_i. \quad (3)$$

For initialization, we use just $\lambda^{[i]}$ as the message from the i th variable node to the j th check node, as indicated by the edges of the Tanner graph B : $\lambda_0^{[i \rightarrow j]} = \lambda^{[i]} \forall \{i, j : [\mathbf{H}]_{ji} = 1\}$. The iteration counter l is now initialized to 1. On the first half-iteration, we compute messages to be sent from the j th check node to the adjacent variable nodes as

$$\lambda_l^{[i \leftarrow j]} = 2 \tanh^{-1} \left(\prod_{k \in \mathcal{N}(j) \setminus i} \tanh \frac{\lambda_{l-1}^{[k \rightarrow j]}}{2} \right). \quad (4)$$

Now, during the second half-iteration, we proceed to sum incoming messages at the i th variable node to be sent back to the adjacent check nodes as

$$\lambda_l^{[i \rightarrow j]} = \lambda^{[i]} + \sum_{k \in \mathcal{N}(i) \setminus j} \lambda_l^{[i \leftarrow k]}. \quad (5)$$

If the early termination logic detects a codeword or the iteration counter exceeds the maximum allowed count, we stop the iteration loop. Otherwise, to continue iterating, we increment l and jump to (4). Upon the completion of the iterations, each symbol decision $\hat{t}_l^{[i]}$ is set to the sign of the expression

$$\hat{t}_l^{[i]} = \lambda^{[i]} + \sum_{k \in \mathcal{N}(i)} \lambda_l^{[i \leftarrow k]}. \quad (6)$$

Assuming no cycles in B , (6) is equivalent to $\ln \frac{P(\mathbf{r} | t_i = +1)}{P(\mathbf{r} | t_i = -1)}$, the MAP decision statistic.

The conditional mean and variance of the LLRs from the AWGN channel (3) are clearly

$$m_\lambda \triangleq \mathbb{E}[\lambda^{[i]} | t_i = +1] = \frac{2}{\sigma^2} = 4 \frac{E_s}{N_0} = 4 \frac{RE_b}{N_0} \quad \text{and} \quad (7)$$

$$\text{VAR}[\lambda^{[i]} | t_i = +1] = \frac{4}{\sigma^2} = 2m_\lambda. \quad (8)$$

D. SPA without Saturation

Direct implementation of (4) yields numerical problems at high LLRs. Double-precision floating-point (64-bit IEEE 754) computer computations maintain 53-bits of precision, while the remaining bits are for the sign and exponent. Thus, such a computer implementation results in $\tanh(x/2)$ being rounded to 1 for any $x > 38.1230949 = 55 \ln 2$.

As an argument of ± 1 will cause the \tanh^{-1} function to overflow, protection from high magnitude LLRs must be added to (4) or (5) to ensure numerical integrity or an alternate solution not using \tanh^{-1} must be found. Thus, preventing \tanh^{-1} overflow by limiting LLRs will result in a maximum producible LLR magnitude. This is what we refer to as “saturating” in a log-domain SPA decoder throughout this paper. Our examination of published error floor results suggests that LLR saturation is commonly employed.

Our non-saturating decoder simulation is based on an exact pairwise check node reduction [34], [35]. Letting $\mathcal{N}(j) \setminus i = \{k_1, k_2\}$, the following exactly equals (4).

$$\begin{aligned} \lambda_l^{[i \leftarrow j]} &= \text{sign } \lambda_{l-1}^{[k_1 \rightarrow j]} \cdot \text{sign } \lambda_{l-1}^{[k_2 \rightarrow j]} \\ &\min \left(\left| \lambda_{l-1}^{[k_1 \rightarrow j]} \right|, \left| \lambda_{l-1}^{[k_2 \rightarrow j]} \right| \right) \\ &+ \ln \left(1 + \exp \left[- \left| \lambda_{l-1}^{[k_1 \rightarrow j]} + \lambda_{l-1}^{[k_2 \rightarrow j]} \right| \right] \right) \\ &- \ln \left(1 + \exp \left[- \left| \lambda_{l-1}^{[k_1 \rightarrow j]} - \lambda_{l-1}^{[k_2 \rightarrow j]} \right| \right] \right) \end{aligned} \quad (9)$$

The check node re-formulation in (9) contains no possibility of overflow for any possible LLR, which for double-precision floating-point extends to a magnitude of approximately 1.79×10^{308} . The only potential for overflow in the SPA is now just the addition operation within (5) at extremely high LLRs.

The computer implementation of (9) does not necessarily have a significant impact on simulation speed. A single hyperbolic tangent evaluation consumes nearly the same number of CPU cycles as four exponential or logarithmic evaluations on a modern processor. The most significant impact is that (9) forces us to organize computations pairwise. Hu et al. present a substantial speed improvement by organizing computation pairs onto a trellis by the forward-backward algorithm, which computes *all* the output messages of the check node [34].

The next level of speed optimization is to approximate the $\ln(1 + \exp(-|x|))$ operations, for which results lie in the interval $(\ln 1, \ln 2]$. There has been significant work in this area, much of it focused towards hardware implementation [23], [34]. We have adopted the following two-piece linear approximation [36]

$$\ln(1 + e^{-|x|}) \approx \begin{cases} 0.6 - 0.24|x|, & \text{if } |x| < 2.5 \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

We have noted losses of about 0.02 dB in the waterfall region of the FER curve for the (2640, 1320) Margulis code with this approximation, while the simulation runs nearly 4 times faster. As the LLRs grow large, the approximation error of (10) becomes relatively insignificant.

It should be noted that Sun developed a different approximation for the check node update (4) based upon different expressions, which he used for large LLRs in his log-domain SPA simulation [1], [19]. Also, as LLRs get very large, the SPA approaches the Min-Sum Algorithm (MSA) as the two logarithmic terms of (9) become relatively small. In the MSA, the check node update (4) of the SPA is replaced with

$$\lambda_l^{[i \leftarrow j]} = \min_{k \in \mathcal{N}(j) \setminus i} \left| \lambda_{l-1}^{[k \rightarrow j]} \right| \cdot \prod_{k \in \mathcal{N}(j) \setminus i} \text{sign } \lambda_{l-1}^{[k \rightarrow j]}. \quad (11)$$

E. Trapping Sets & Absorbing Sets

Definition 1 (Richardson [8]). A *trapping set* is an error-prone set of variable nodes \mathcal{T} that depends upon the decoder's input space and decoding algorithm. Let $\hat{t}_l^{[i]}(\mathbf{r})$ denote the i th output symbol at the l th iteration given the decoder input vector \mathbf{r} . We say that symbol i is *eventually correct* if there exists L such that, for all $l \geq L$, $\hat{t}_l^{[i]}(\mathbf{r})$ is correct. If the set of symbols that is not eventually correct is not the empty set, we call it

a trapping set, $\mathcal{T}(\mathbf{r})$. It is called an (a, b) trapping set if the set has $a = |\mathcal{T}|$ variable nodes and the induced subgraph $B_{\mathcal{T}}$ contains exactly b check nodes of odd degree.

From here on this paper will use the term *trapping set*, unqualified, to mean the trapping sets defined above with respect to the AWGN channel decoded by a log-domain SPA decoder with saturating LLRs. Note, that while near-codewords are distinct from codewords, the trapping set definition includes codewords.

Definition 2. A Tanner subgraph or trapping set is called *elementary* if all the check nodes are of degree one or two [15].

Elementary trapping sets are simple enough to model with linear or nearly-linear systems and also account for most of the error floors seen in practice. In [8], [12], [14], [15], [37]–[39], the authors observe that the majority of trapping sets contributing to the error floors of belief propagation decoders are elementary trapping sets.

Example 1. Four sample elementary Tanner subgraphs are shown in Fig. 2 for a code of variable-degree three. These are all trapping sets if they satisfy the “not eventually correct” condition of Definition 1.

We will define absorbing sets as they dominate the error floor of SPA decoders with saturating LLRs. Absorbing sets are a particular type of near-codeword or trapping set. The fully absorbing definition adds conditions beyond the subgraph itself to the remaining Tanner graph.

Definition 3 (Dolecek et al. [13]). An (a, b) *absorbing* set of the Tanner graph, is a set of variable nodes of cardinality a which induce a subgraph with exactly b odd degree check nodes, in which all variable nodes neighbor strictly more even-degree check nodes than odd-degree check nodes within the subgraph. Moreover, in a *fully absorbing set* every variable node outside of the induced subgraph neighbors strictly fewer odd-degree check nodes of the subgraph than other check nodes of the Tanner graph.

Fully absorbing sets are stable structures during decoding using the bit-flipping algorithm [13]. This results from every variable node of the set neighboring strictly more checks which reinforce the incorrect value than checks working to correct the variable node.

III. STATE-SPACE MODEL

In this section we develop a linear system model which closely approximates the non-linear message passing of the log-domain SPA that we presented in the prior section, as applied to an elementary trapping set.

The state-space model was introduced in [1], [19] to analyze elementary trapping sets. Identifying and modeling trapping sets is of interest so as to explain the observed phenomenon of the trapping sets' variable nodes decoding incorrectly, while the variable nodes outside of the set are eventually decoded correctly by a log-domain SPA decoder with saturation.

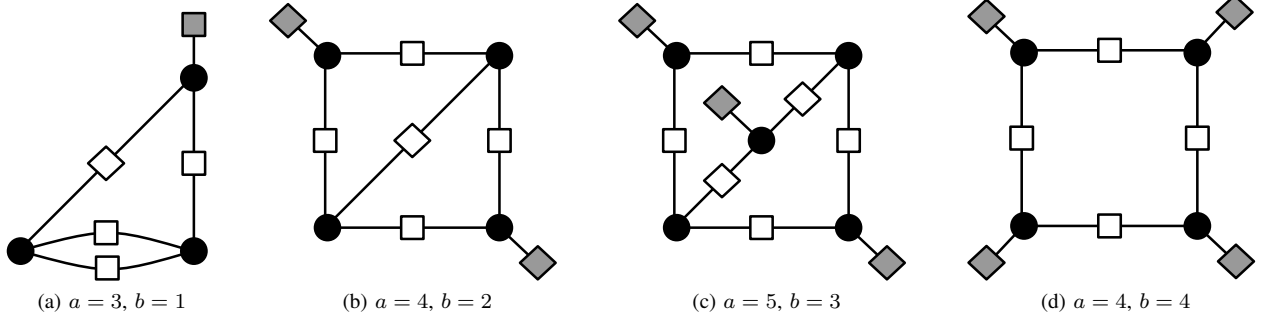


Fig. 2. Three sample elementary Tanner subgraphs of a code with $d_v = 3$. Unsatisfied, degree-one check nodes are shown shaded.

Assumption 2. All trapping sets of interest are elementary and contain more than one variable node. We do not consider subgraphs with a single variable node as they would not have states in the state-space model. Since this is a linear block code, we assume for simplicity that the all-zero codeword has been sent.

First, we review the failure state of an elementary trapping set. Let the variable nodes of the trapping set $\mathcal{S} \subset V$ induce the Tanner subgraph $B_{\mathcal{S}}$. In later iterations the condition is reached where the variable nodes \mathcal{S} within the trapping set are in error and those variable nodes outside the trapping set (i.e., $V \setminus \mathcal{S}$) are correct. In this condition, the subgraph's check nodes of degree one are unsatisfied and those of degree two are satisfied.

The vector of messages from the degree-one (i.e., unsatisfied) check nodes $\lambda_l^{(ex)}$ at iteration l are taken to be stochastic system inputs to the state-space model and are therefore modeled separately. We will use density evolution and simulation techniques, described in later sections, to model this input vector. The other system input vector is the intrinsic information λ provided by the channel to be used in variable node updates every iteration. From the AWGN channel model, we know that each element of λ is an i.i.d. Gaussian random variable produced by the channel and scaled to be an LLR as shown in (3). These system inputs will be treated as column vectors of LLRs; the vector λ has a entries and the vector $\lambda_l^{(ex)}$ has b entries for an (a, b) trapping set.

A. Check Node Gain Model

Sun's linear model included the asymptotic approximation that every degree-two (i.e., satisfied) check node output message is equal to the input message [1]. This is based on the conditions of correct variable nodes and very high LLR values outside of the trapping set. Schlegel and Zhang introduced more accuracy to this model by applying a multiplicative gain g_l at the degree-two check nodes, where $0 < g_l \leq 1$ [20]. This gain models the effect of the $d_c - 2$ external variable nodes lowering the magnitude of LLR messages as they pass through the degree-two check nodes. This is illustrated in Fig. 3. This approach adds accuracy to modeling the early iterations, and after several iterations these gains approach 1.

Assumption 3. We will assume that the external inputs to the trapping set's degree-two check nodes are i.i.d. This

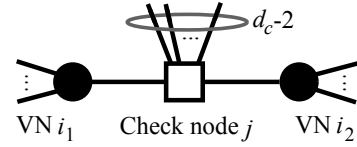


Fig. 3. Illustration of a check node of degree two in the trapping set and degree d_c in the Tanner graph.

is generally a reasonable assumption to make as the gain computations are most significant at early iterations. By the iteration count in which significant message correlation is present on the check nodes' inputs, the gains are approximately 1.

Schlegel and Zhang compute the mean gain \bar{g}_l corresponding to each iteration count l . Thus, the gain model turns the state updates into a time-variant process. We have tried introducing a gain variance to the model, but found the effect to be very small, so we present their mean gain model, for which we take g_l to be

$$g_l \triangleq \prod_{k=1}^{d_c-2} \tanh\left(\frac{\lambda_{l-1}^{[k \rightarrow j]}}{2}\right), \quad (12)$$

for a check node of degree d_c . Then the expected value of the check node's gain over channel realizations is

$$\bar{g}_l = \mathbb{E}_{\lambda} \left[\prod_{k=1}^{d_c-2} \tanh\left(\frac{\lambda_{l-1}^{[k \rightarrow j]}}{2}\right) \right]. \quad (13)$$

In practice, we will use the following simplification with respect to (13)

$$\bar{g}_l = \mathbb{E}_{\lambda} \left[\tanh\left(\frac{\lambda_{l-1}^{[k \rightarrow j]}}{2}\right) \right]^{d_c-2}, \quad (14)$$

which uses Assumption 3 and satisfies $0 < \bar{g}_l \leq 1$.

The application of gain at the check node may be justified using a Taylor series expansion of (4) that begins with letting

$$\lambda_{out} \triangleq f(\lambda) = 2 \tanh^{-1} \left[\tanh\left(\frac{\lambda}{2}\right) g \right], \quad (15)$$

where g is the check node's gain (12) that we are interested in introducing, λ is the check node's input LLR from the trapping

set and λ_{out} is the check node's output LLR on the other edge within the trapping set. Noting that $f(0) = 0$ and that the first two partial derivatives of (15) evaluated at $\lambda = 0$ are

$$\left. \frac{\partial f(\lambda)}{\partial \lambda} \right|_{\lambda=0} = g \quad \text{and} \quad \left. \frac{\partial^2 f(\lambda)}{\partial \lambda^2} \right|_{\lambda=0} = 0,$$

we produce the approximation $\lambda_{out} \approx g\lambda$ based on the first three terms of the Taylor series about $\lambda = 0$. The approximation is good for $|\lambda| \leq 2 \tanh^{-1}(g)$. The exact expression for λ_{out} (15) saturates at an output of $\pm 2 \tanh^{-1}(g)$, while the approximation is linear in λ . Since we expect the LLRs outside of the trapping set to grow quite fast, we believe this to be an adequate approximation.

We now extend the check node gain model to include inversions through the degree-two check nodes caused by erroneous messages from outside of the trapping set. This occurs primarily during early iterations and is illustrated in Fig 3. When an odd number of the $d_c - 2$ messages from outside of the trapping set into a degree-two check node within the trapping set are erroneous, then that check node will invert the polarity of the LLR message sent from variable node i_1 to i_2 , as well as the message sent in the opposite direction.

We will present a simplification of the method used by Schlegel and Zhang to account for these inversions [20]. Moreover, they present a small rise in their predicted error floor instead of lowering it as we find with our technique. Schlegel and Zhang injected a stochastic cancellation signal into the state update equations, while we choose to modify the mean check node gains.

During the first iteration, messages from variable nodes may contain inversions due to channel errors. The probability that a specific input to the check node contains an inversion during iteration $l = 1$ is just the uncoded symbol error rate

$$P_{e,1} = Q\left(\sqrt{\frac{2RE_b}{N_0}}\right). \quad (16)$$

Thus, utilizing Assumption 3, the probability of a polarity inversion in a specific check node is given at iteration l by

$$P_{i,l} = \sum_{k=1}^{\lfloor (d_c-1)/2 \rfloor} \binom{d_c-2}{2k-1} P_{e,l}^{2k-1} (1 - P_{e,l})^{d_c-1-2k}, \quad (17)$$

which is the probability of an odd number of errors in the $d_c - 2$ input messages at iteration l .

For additional iterations we can use density evolution [40], [41] to predict an effective E_b/N_0 at the output of the variable nodes and reuse equations (16) and (17). When there is a channel inversion the output message magnitude will likely be very low, as suggested in [20]. Hence, to model this we will equate random channel inversions to randomly setting the check node gain to zero with probability $P_{i,l}$. We introduce the check node's *modified mean gain* as simply

$$\bar{g}'_l = \bar{g}_l(1 - P_{i,l}). \quad (18)$$

For an LDPC code with regular check node degree d_c , the scalar gain \bar{g}'_l of (18) may be applied at each iteration l to model the external influence of the subgraph's degree-two check nodes. For an LDPC code with irregular check degree,

we have a few options. If the check node degree spread is small, we may generalize (13) and (18) by also taking the expectation over the degree distribution of all the degree-two check nodes in the subgraph. Alternately, we may maintain several versions of gain \bar{g}'_l , one for each check-degree that appears in the subgraph.

Assumption 4. For simplicity in modeling and analysis, we assume that all the degree-two check nodes in the subgraph may be modeled by the modified mean gain \bar{g}'_l .

B. Linear State-Space Model

Assumption 5. In deriving the model we will assume that the messages within the trapping set under study have little impact on the rest of the Tanner graph.

The state-space modeling equations are shown below for input column vectors $\lambda_l^{(ex)}$ and λ and for output column vector $\tilde{\mathbf{t}}_l$, whose entries $\tilde{t}_l^{[i]}$ are the log-domain SPA estimate of the sign of the i th transmitted binary antipodal symbol, at iteration l , i.e., $\tilde{t}_l^{[i]} = \text{sign}(\tilde{t}_l^{[i]})$ [1], [20].

$$\mathbf{x}_0 = \mathbf{B}\lambda \quad (19)$$

$$\mathbf{x}_l = \bar{g}'_l \mathbf{A} \mathbf{x}_{l-1} + \mathbf{B}\lambda + \mathbf{B}_{ex} \lambda_l^{(ex)} \quad \text{for } l \geq 1 \quad (20)$$

$$\tilde{\mathbf{t}}_l = \bar{g}'_l \mathbf{C} \mathbf{x}_{l-1} + \lambda + \mathbf{D}_{ex} \lambda_l^{(ex)} \quad \text{for } l \geq 1 \quad (21)$$

The central part of the state-space model, of course, is the updating of the state \mathbf{x}_l . The state vector \mathbf{x}_l represents the vector of LLR messages sent along the edges from the subgraph's variable nodes towards the degree-two check nodes. The state is updated once per full iteration. One may consider that $\bar{g}'_l \mathbf{x}_{l-1}$ represents the LLR messages after passing through the degree-two check nodes during the first half-iteration, and $\bar{g}'_l \mathbf{A} \mathbf{x}_{l-1}$ represents their contribution to the variable node update (5) during the second half-iteration of iteration l . Since ad_v equals the number of edges of the variable-regular Tanner subgraph of a variable nodes, the number of states in our model of an elementary subgraph is $m = ad_v - b$, that is, just the m edges incident on the degree-two check nodes. Note that m must be an even integer.

The $m \times a$ matrix \mathbf{B} and the $m \times b$ matrix \mathbf{B}_{ex} are used to map λ and $\lambda_l^{(ex)}$, respectively, to the appropriate entries of the state vector to match the set of variable node update equations (5). Since every variable node has exactly one element from λ participating as shown in (5), \mathbf{B} has a single 1 in every row, and is zero otherwise. The number of 1 entries per row of \mathbf{B}_{ex} corresponds to the number of degree-one check nodes adjacent to the corresponding variable node, which may be none. Like all the matrices appearing in the state-space equations, they are $(0, 1)$ -matrices.

The $m \times m$ matrix \mathbf{A} describes the dependence of the state update upon the prior state vector. Each nonzero entry $[\mathbf{A}]_{ij} = 1$ indicates the j th edge of the prior iteration contributes to the variable node update computation of the i th edge. Thus, the matrix \mathbf{A}^T may be taken as the adjacency matrix associated with a simple digraph of order m which describes the state-variable update relationships. Given our

d_v -variable-regular codes, the row weight of \mathbf{A} and the row weight of \mathbf{B}_{ex} will sum to $d_v - 1$ for every row.

Finally, the $a \times m$ matrix \mathbf{C} and the $a \times b$ matrix \mathbf{D}_{ex} are used to map \mathbf{x}_{l-1} and $\lambda_l^{(ex)}$ entries, respectively, to the corresponding entry of the soft output decision vector $\tilde{\mathbf{t}}_l$. Thus, $\tilde{g}'_l \mathbf{C} \mathbf{x}_{l-1} + \mathbf{D}_{ex} \lambda_l^{(ex)}$ forms a vector representation of the $\sum_{k \in \mathcal{N}(i)} \lambda_l^{[i-k]}$ terms in the soft output equations (6). The row weight of \mathbf{C} and the row weight of \mathbf{D}_{ex} will sum to the variable node degree $d(v_i)$ for the i th row.

If there is a single degree-one check node neighboring every variable node in the subgraph, such as in the (8, 8) trapping set of 802.3an, then (20) degenerates to the case where $\mathbf{B} = \mathbf{B}_{ex}$ and \mathbf{B} has regular column weight as derived in [20]. Furthermore, this degenerate case produces only \mathbf{A} matrices which have uniform row and column sums and only $r = d_v - 2$. Thus, our development will be significantly more general than [20].

Again, note that the check node gains form our approximate model for the behavior of degree-two check nodes within the trapping set. One drawback to any linear system approach is that saturation cannot be introduced to the states variables. Any saturation effects must be applied to linear system inputs.

Example 2. For the (4, 2) trapping set with $d_v = 3$, we label with integers all the edges into every degree-two check node as depicted in Fig. 4a. Now, we can form the matrices used to make the linear system. We have numbered the edges in Fig. 4a to correspond to the order that the edges will appear in the state vector \mathbf{x}_l . The matrix \mathbf{A} describes the updating relationship of the state vector in a full iteration of the SPA. Since edge 1 depends only on edges 6 and 9 during one iteration, we fill the first row of \mathbf{A} to indicate such. All the matrices associated with the trapping set of Fig. 4 are given below. Additionally, treating the matrix \mathbf{A}^T as an adjacency matrix we get the corresponding simple digraph shown in Fig. 4d.

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (22)$$

$$\mathbf{B}_{ex} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \quad \mathbf{C} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad \mathbf{D}_{ex} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$

If we remove the gains \tilde{g}'_l from the linear state-space equations, we have a linear time-invariant (LTI) system for which we can write a transfer function. Let the unilateral Z-transform of the state \mathbf{x}_l be $\mathcal{Z}\{\mathbf{x}_l\} = \mathbf{X}(z) = \sum_{k=0}^{\infty} \mathbf{x}_k z^{-k}$ and the Z-transform of input vector $\lambda_l^{(ex)}$ be denoted $\mathbf{\Lambda}_{ex}(z)$. Note that the other input vector λ is not a function of iteration number. Using $\mathcal{Z}\{\mathbf{x}_{l+1}\} = z\mathbf{X}(z) - z\mathbf{x}_0$ and taking the Z-transform of (20), we have [42]

$$\mathbf{X}(z) - \mathbf{x}_0 = \mathbf{A}\mathbf{X}(z)z^{-1} + \frac{\mathbf{B}\lambda z^{-1}}{1 - z^{-1}} + \mathbf{B}_{ex}\mathbf{\Lambda}_{ex}(z).$$

Solving for $\mathbf{X}(z)$ and noting that $\mathbf{x}_0 = \mathbf{B}\lambda$, yields

$$\mathbf{X}(z) = (\mathbf{I} - \mathbf{A}z^{-1})^{-1} \left[\frac{\mathbf{B}\lambda}{1 - z^{-1}} + \mathbf{B}_{ex}\mathbf{\Lambda}_{ex}(z) \right]. \quad (23)$$

Now, the complete LTI system can be expressed in the Z-domain. Let the soft output vector $\tilde{\mathbf{t}}_l$ of the system be denoted in the Z-domain by $\tilde{\mathbf{T}}(z)$, and letting $\tilde{\mathbf{t}}_0 = \lambda$. Then, plugging (23) into the Z-transform of (21) yields

$$\tilde{\mathbf{T}}(z) = \left[\mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1} \mathbf{B} + \mathbf{I} \right] \frac{\lambda}{1 - z^{-1}} + \left[\mathbf{C}(z\mathbf{I} - \mathbf{A})^{-1} \mathbf{B}_{ex} + \mathbf{D}_{ex} \right] \mathbf{\Lambda}_{ex}(z). \quad (24)$$

This makes evident the importance of the eigenvalues of \mathbf{A} . As the eigenvalues are the roots of the characteristic equation $\det(\mathbf{A} - \mu\mathbf{I}) = 0$, they correspond directly to the poles of the discrete-time LTI system in (24). We find that for the trapping sets addressed herein, the dominant poles are on or outside the unit circle. Hence, the LTI system is marginally stable or (more often) unstable.

We wish to take the recursive state-update equation (20) and develop it into an expression without feedback. The first two iterations are easy to express term-by-term, as

$$\begin{aligned} \mathbf{x}_1 &= \tilde{g}'_1 \mathbf{A} \mathbf{B} \lambda + \mathbf{B} \lambda + \mathbf{B}_{ex} \lambda_1^{(ex)} \quad \text{and} \\ \mathbf{x}_2 &= \tilde{g}'_1 \tilde{g}'_2 \mathbf{A}^2 \mathbf{B} \lambda + \tilde{g}'_2 \mathbf{A} \mathbf{B} \lambda + \mathbf{B} \lambda \\ &\quad + \tilde{g}'_2 \mathbf{A} \mathbf{B}_{ex} \lambda_1^{(ex)} + \mathbf{B}_{ex} \lambda_2^{(ex)}. \end{aligned}$$

Generalizing to an arbitrary iteration $l > 0$, we have

$$\mathbf{x}_l = \mathbf{A}^l \mathbf{B} \lambda \prod_{j=1}^l \tilde{g}'_j + \sum_{i=1}^l \mathbf{A}^{l-i} \left(\mathbf{B} \lambda + \mathbf{B}_{ex} \lambda_i^{(ex)} \right) \prod_{j=i+1}^l \tilde{g}'_j. \quad (25)$$

C. Graphical Assumptions & Interrelationships

We have already noted that we will analyze binary codes that are described by variable-regular Tanner graphs of variable-degree $d_v \geq 3$ for binary antipodal transmission over the AWGN channel. Also, the trapping sets that we characterize must be elementary and have more than one variable node. This subsection notes some further assumptions we wish to place on the trapping sets and describes the interrelationships among several useful graphical descriptions of the trapping set: the Tanner subgraph, a simpler undirected multigraph, and the directed graph that corresponds directly to the state update model.

Lemma 3. *There exists a bijective map between the set of d_v -variable-regular elementary Tanner subgraphs (up to a relabeling of the check nodes) and the set of multigraphs $G = (V, F)$ with vertex degrees upper bounded by d_v , i.e., $d(v_i) \leq d_v \forall v_i \in V$.*

Proof: Given the elementary Tanner subgraph $B_S = (\mathcal{S}, \mathcal{N}(\mathcal{S}), E_S)$, each variable node $v \in \mathcal{S}$ becomes a vertex $v \in V$ of G , that is $V = \mathcal{S}$, and the degree-one check nodes are discarded. The check nodes of degree two become the edges of G , each joining a pair of vertices. Alternately, let parity check submatrix \mathbf{H}_S describe B_S , then $\mathbf{A}(G) = \mathbf{H}_S^T \mathbf{H}_S - d_v \mathbf{I}$.

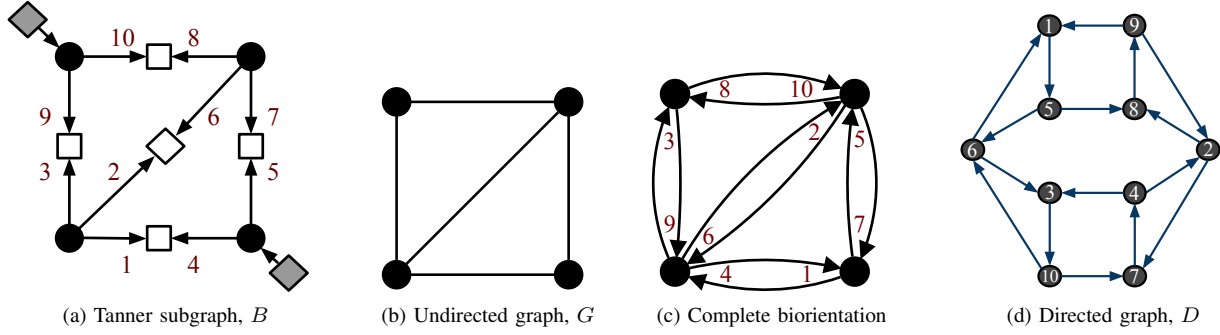


Fig. 4. Several graphical descriptions of the $(4, 2)$ trapping set with model's states numbered. Arrows have been added to the Tanner subgraph in (a) only to emphasize the direction of state variable flow. The digraph in (c) shows the complete biorientation of G .

Given the multigraph G , each vertex $v \in V$ of G becomes a variable node $v \in \mathcal{S}$ of $B_{\mathcal{S}}$. So, again $V = \mathcal{S}$. Each edge of G is replaced by a degree-two check node (with an arbitrary label) and two edges in $B_{\mathcal{S}}$. Finally, degree-one check nodes are attached with single edges to variable nodes as needed until every variable node has d_v neighbors. ■

In creating multigraph G , we have a more basic description of the potential elementary trapping set. This is illustrated in creating Fig. 4b from Fig. 4a. Either one of these graphs is sufficient to characterize the iteration-to-iteration dependencies among the state variables in our model. We allow for parallel edges in G of Lemma 3 to map Tanner graphs with cycles of length 4, the smallest permitted in a Tanner graph. Cycles of length k in $B_{\mathcal{S}}$ map to cycles of length $k/2$ in G . The 4-cycles in the Tanner graph are often avoided by code designers.

$$\begin{aligned} \text{order}(G) &= a & \text{and} \\ \text{size}(G) &= \frac{ad_v - b}{2} \end{aligned}$$

Example 3. The multigraph corresponding to the Tanner subgraph of Fig. 2d is the cycle graph of order 4, C_4 .

Assumption 6. Tanner subgraphs of interest and their associated multigraphs are connected.

Those trapping sets described by disconnected Tanner subgraphs that we have ruled out by Assumption 6 can instead be analyzed component by component. The main reason for this simplification is to reduce the number of cases in which the state update expression contains a reducible \mathbf{A} matrix.

Assumption 7. We further assume that the variable nodes within the Tanner subgraphs contain from zero to $d_v - 2$ adjacent degree-one check nodes. Equivalently, we assume that the associated multigraphs are leafless; that is, they do not contain vertices of degree one.

A leaf in the multigraph maps to a variable node in the Tanner subgraph neighboring one degree-two check node and $d_v - 1$ degree-one check nodes. Thus, we require here that every variable node in the Tanner subgraph must neighbor at least two degree-two check nodes. The multigraphs allowed by Assumptions 6 and 7 will be connected leafless graphs, containing one or more cycles. For consideration of graphs with leaves, see Appendix B.

The conditions described here, Sun termed “simple trapping sets” in [1]. Our Assumption 7 is equivalent to elementary absorbing sets for codes of variable-degree three. However, our Assumption 7 is less restrictive in some ways than absorbing sets for $d_v \geq 4$ as we allow for more degree-one check nodes per variable node.

Finally, we describe how to create the simple digraph $D = (Z, A)$ from an undirected multigraph $G = (V, F)$ which meets Assumptions 6 and 7. This is illustrated in creating Fig. 4d from Fig. 4b. Each edge of a connected leafless multigraph G corresponds to two vertices in D , representing the two directions of LLR message flow in the original Tanner graph. That is, the number of vertices m in D is simply

$$m = \text{order}(D) = 2 \text{ size}(G) = ad_v - b. \quad (26)$$

The digraph D is a representation of message flow out of the variable nodes for one full iteration of the SPA, (5) in particular. Let edge $f_i = (v_j, v_k) \in F$ map to vertices $z_i, z_{i'} \in Z$, with z_i representing the direction of f_i flowing from v_j to v_k and $z_{i'}$ representing the other direction. No arcs in D join z_i to $z_{i'}$. Arcs initiating in z_i are directed to vertices corresponding to other edges in G flowing out of v_k and arcs terminating in z_i are directed from vertices corresponding to other edges in G flowing into v_j . Hence,

$$d_i^+ = \deg(v_k) - 1 = d_{i'}^-, \quad (27)$$

$$d_{i'}^- = \deg(v_j) - 1 = d_i^+, \quad \text{and} \quad (28)$$

$$\text{size}(D) = \sum_{z_i \in Z} d_i^- = \sum_{v_j \in V} d(v_j) [d(v_j) - 1]. \quad (29)$$

With respect to a Tanner subgraph that meets Assumption 7, induced from a code of variable-degree three, the size of the associated digraph simplifies to $6a - 4b$. In our $(4, 2)$ example, the digraph of Fig. 4d is order 10 and size 16.

We now describe a second version of the construction of the simple digraph that relates closely to line graphs. From the undirected multigraph G , shown in Fig. 4b, we create the multidigraph D' shown in Fig. 4c by replacing each edge in G with a pair of arcs, one in each direction. The multidigraph D' is known as the complete biorientation of G . The adjacency matrix of D' is $\mathbf{A}(D') = \mathbf{A}(G) = \mathbf{H}_{\mathcal{S}}^T \mathbf{H}_{\mathcal{S}} - d_v \mathbf{I}$. Next, we create the line digraph $\mathcal{L}(D')$ and delete $m/2$ arc-pairs to form the simple digraph D , shown in Fig. 4d. The arc-pairs to be deleted are incident to the vertex-pairs in $\mathcal{L}(D')$,

which correspond to the arc-pairs in D' that originated from the individual edges in G . In our example, we have deleted arc-pairs in $\mathcal{L}(D')$ between vertices 1 and 4, between vertices 2 and 6, and so on, to create D as shown in Fig. 4d.

All directed walks in D will correspond to walks in G . All walks in G that do not backtrack will correspond to directed walks in D . Backtracking is prohibited here due to the structure of SPA as expressed in (5), which excludes an edge's own incoming LLR from its outgoing LLR computation. It is for this reason that the arc-pairs are deleted from the line digraph.

If undirected multigraph G is regular, then both D' and D are regular. For each k -cycle in G , there are two corresponding directed cycles of length k in D , one for each direction.

The adjacency matrix of D is $\mathbf{A}(D)$, which is an $m \times m$ (0,1)-matrix. The \mathbf{A} used in state space model (20) is equal to $\mathbf{A}^T(D)$ as D describes the flow of messages within our model. This connection will be convenient when discussing the properties of \mathbf{A} .

Example 4. Figs. 2b, 2c, 2d, 2a, and 5c meet Assumptions 6 and 7. Of these, only Fig. 2d has a reducible $\mathbf{A}(D)$ matrix and it may be analyzed in this paper as explained in Appendix A. The graphs of Figs. 5a and 5b fail Assumption 7 and have reducible $\mathbf{A}(D)$ matrices. In the case of Fig. 5a, the graph may first be analyzed without the leaf, and then the leaf considered as explained in Appendix B. The case of Fig. 5b has all zero eigenvalues as no amount of leaf removal creates a satisfactory structure. Thus, it is not analyzed in this paper.

IV. DOMINANT EIGENVALUES & PROBABILITY MODEL

This section will simplify the linear equation used to update the states developed in the prior section to the point that we can easily apply a probability model and predict failure rates for specific trapping sets.

A. Utilizing Dominant Eigenvalues

For analysis of our model, we need to simplify the powers of \mathbf{A} that appear in (25). To do this we will rely on the dominant eigenvalues of the nonnegative $m \times m$ matrix \mathbf{A} and its left eigenvectors. This approach will avoid some of the rough edges of prior analyses. Sun assumed that the matrix \mathbf{A} was diagonalizable [1]. However, our extensive search of potential trapping sets in Appendix C showed a small fraction of them have \mathbf{A} matrices which cannot be diagonalized. Schlegel and Zhang used an asymptotic approximation to \mathbf{A}^i in [20] which we also want to avoid, because we only need to model a small number of iterations for a decoder with saturating LLRs.

Let $\mu_k \in \mathbb{C}$ be an eigenvalue of matrix \mathbf{A} , and let \mathbf{w}_k be the left eigenvector associated with μ_k , such that $\mathbf{w}_k^* \mathbf{A} = \mu_k \mathbf{w}_k^*$, where \mathbf{w}_k^* is the conjugate transpose of column vector \mathbf{w}_k . Then, by induction, for any positive integer i ,

$$\mathbf{w}_k^* \mathbf{A}^i = \mu_k^i \mathbf{w}_k^*. \quad (30)$$

Left-multiplying (25) by \mathbf{w}_k^* and using (30) to simplify, we

derive the scalar quantity

$$\begin{aligned} \mathbf{w}_k^* \mathbf{x}_l &= \mu_k^l \mathbf{w}_k^* \mathbf{B} \lambda \prod_{j=1}^l \bar{g}_j' \\ &+ \sum_{i=1}^l \mu_k^{l-i} \mathbf{w}_k^* \left(\mathbf{B} \lambda + \mathbf{B}_{ex} \lambda_i^{(ex)} \right) \prod_{j=i+1}^l \bar{g}_j'. \end{aligned} \quad (31)$$

We now shift to using a specific left eigenvector of the nonnegative matrix \mathbf{A} , the left eigenvector \mathbf{w}_1 associated with the eigenvalue of maximum modulus r . Dividing the $m \times m$ nonnegative matrix \mathbf{A} into the following two cases, the theory of nonnegative matrices allows us to make certain statements regarding \mathbf{w}_1 and r [27]–[30]:

- 1) Let the nonnegative matrix \mathbf{A} be irreducible. There is a simple eigenvalue r , such that $r = \rho(\mathbf{A})$ and $r > 0$. The associated left eigenvector \mathbf{w}_1 of r is positive. There are no other nonnegative left eigenvectors of \mathbf{A} , except for positive multiples of \mathbf{w}_1 .
- 2) Let the nonnegative matrix \mathbf{A} be reducible. Letting \mathbf{P} be an $m \times m$ permutation matrix, the matrix \mathbf{A} may be symmetrically permuted to $\mathbf{A}' = \mathbf{P} \mathbf{A} \mathbf{P}^T$, where \mathbf{A}' is in block upper triangular form. The block diagonal submatrices \mathbf{A}'_i are either irreducible square matrices or the 1-by-1 zero matrix. The spectrum of \mathbf{A} is thus the union of the individual spectra $\sigma(\mathbf{A}'_i)$. Appendix A explains that the reducible cases allowed by Assumptions 6 and 7 have all their eigenvalues located on the unit circle, each with multiplicity two. For these cases, we can associate the positive left eigenvector $\mathbf{w}_1 = [1, 1, \dots, 1]^T$ with the eigenvalue $r = 1$. Of course there exist other nonnegative eigenvectors.

Thus, given our prior assumptions, we may associate a positive left eigenvector \mathbf{w}_1 with the real eigenvalue r , and this r is real with magnitude greater than or equal to that of all other eigenvalues.

We now define the error indicator $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$. We define β_l this way to create a scalar indicator of trapping set error. Consider β_l as the scaled projection of the state vector \mathbf{x}_l onto the positive vector \mathbf{w}_1 . Since the state vector \mathbf{x}_l contains the internal messages of the trapping set, the projection onto a positive vector indicates the trapping set's messages are generally either in the positive (correct) direction or negative (erroneous) direction.

Thus β_l is only approximately indicative of the variable nodes' decisions. Especially in the case where the system is inherently unstable, $r > 1$, the state variables will tend to all move toward positive or negative infinity and dominate the soft output expression (21). The domination of (21) by \mathbf{x}_{l-1} is obvious in the case that the $\lambda_i^{(ex)}$ terms are saturated to finite values. Even when this is not the case, the \mathbf{x}_{l-1} terms still dominate the soft output expression (21) as the state update equation (20) is much like (21), but (21) puts even more weight on the states.

Example 5. We continue the on-going example. The non-negative matrices \mathbf{A} describing the state update operations of Figs. 2a and 2b are both primitive, with $r = 1.6956$ and

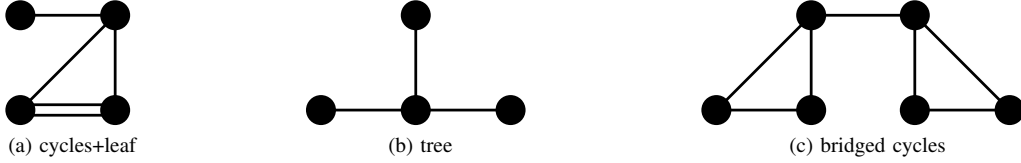


Fig. 5. Three sample diagrams of undirected multigraphs.

1.5214, respectively. The nonnegative matrix \mathbf{A} describing Fig. 2c is imprimitive with $h = 4$ and $r = \sqrt{2}$. This means the four eigenvalues on the spectral circle are $\pm\sqrt{2}$ and $\pm\sqrt{2}i$. One may find that $h = 4$ by visual inspection of Fig. 2c, noting that every cycle length (measured in full iterations) is divisible by four. The matrix \mathbf{A} describing the state update of Fig. 2d is reducible with $r = 1$.

Our error indicator expression simplifies if we rescale β_l by a positive constant to $\beta'_l \triangleq \beta_l / \left(r^l \prod_{j=1}^l \bar{g}_j' \right)$, so

$$\beta'_l = \mathbf{w}_1^T \mathbf{B} \lambda + \sum_{i=1}^l \frac{\mathbf{w}_1^T (\mathbf{B} \lambda + \mathbf{B}_{ex} \lambda_i^{(ex)})}{r^i \prod_{j=1}^i \bar{g}_j'}. \quad (32)$$

This expression is similar to, but more general than (1) in [20], which was derived for a specific degenerate trapping set as previously described. Thus, (32) allows us to consider a much wider class of trapping sets, such as the dominant trapping sets of the Margulis code.

B. Probability of Error Model

The expression for β'_l in (32) is a linear combination of stochastic vectors λ and $\lambda_i^{(ex)}$. Since elements of λ are i.i.d. Gaussian, linear operations on λ will produce a Gaussian distribution. Several authors [20], [40] have used the approximation that the check node output LLRs, such as $\lambda_i^{(ex)}$, are Gaussian, too. The central limit theorem implies that the distribution of a linear combination of many independent messages will be nearly Gaussian, even if the check node output LLRs are only approximately Gaussian.

Assumption 8. We assume that β'_l has a Gaussian distribution.

Under this assumption, the probability of the failure event, $\xi(\mathcal{S})$, corresponding to trapping set \mathcal{S} , at iteration l , is then simply

$$P\{\xi(\mathcal{S})\} = P\{\beta'_l < 0\} = Q\left(\frac{\mathbb{E}[\beta'_l]}{\sqrt{\text{VAR}[\beta'_l]}}\right) \quad (33)$$

If $\{\mathcal{S}_i\}$ enumerates all potential trapping sets, the union bound provides an upper bound on the error floor of the codeword error rate

$$P_f \leq \sum_i P\{\xi(\mathcal{S}_i)\}. \quad (34)$$

The codeword error rate of (34) is commonly called the frame error rate (FER) or block error rate. Next, we wish to express the error floor as an information bit error rate (BER). Letting \hat{a}_i represent the maximum number of information bits associated

with trapping set failure \mathcal{S}_i for the specific encoding technique used, we may express the BER union bound as

$$P_b \leq \sum_i \frac{\hat{a}_i}{k} P\{\xi(\mathcal{S}_i)\}, \quad (35)$$

where k is the number of information bits per codeword. When the codeword is encoded *systematically* the codeword bit locations are partitioned between information bits and parity bits. For a systematic encoding, in which the bit error positions are spread uniformly with no preference to information and parity locations, we may state $\mathbb{E}[\hat{a}_i] = a_i k / n$, where a_i is number of variable nodes in trapping set \mathcal{S}_i and n is the block length of the codeword. In this case, (35) simplifies to

$$P_b \leq \sum_i \frac{a_i}{n} P\{\xi(\mathcal{S}_i)\}. \quad (36)$$

C. Codewords

In the case of elementary trapping sets that are also codewords, we let $b = 0$ and $\mathbf{B}_{ex} = 0$. Using (7), the expected value of (32) simplifies to

$$\mathbb{E}[\beta'_l] = m_\lambda \left(1 + \sum_{i=1}^l \frac{1}{r^i \prod_{j=1}^i \bar{g}_j'} \right) \sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k. \quad (37)$$

Using the variance (8) and the independence property of the LLRs from the channel, the variance of (32) is

$$\text{VAR}[\beta'_l] = 2m_\lambda \left(1 + \sum_{i=1}^l \frac{1}{r^i \prod_{j=1}^i \bar{g}_j'} \right)^2 \sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k^2. \quad (38)$$

Now we can see that the failure probability (33) of the codeword can be stated as

$$P\{\xi(\mathcal{S}_i)\} = Q\left(\sqrt{\frac{2RE_b}{N_0}} \frac{\sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k}{\sqrt{\sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k^2}}\right). \quad (39)$$

This reduces further as the eigensystem of \mathbf{A} for codewords is rather simple. Every row sum and column sum of the matrix \mathbf{A} is $d_v - 1$, so the spectral radius is $r = d_v - 1$. The positive left eigenvector associated with r is proportional to the all-ones vector, $\mathbf{w}_1 \propto [1, 1, \dots, 1]^T$. Also, \mathbf{B} has uniform row weight one and column weight d_v . Therefore, $(\mathbf{w}_1^T \mathbf{B})_k = d_v \forall k \in \{0, 1, \dots, a\}$. We recognize that a is just the Hamming weight w_H in the context of codewords, so (39) simplifies to the more recognizable

$$P\{\xi(\mathcal{S}_i)\} = Q\left(\sqrt{\frac{2RE_b}{N_0}} w_H\right), \quad (40)$$

independent of iteration count l .

D. Non-codewords

To further understand the general behavior of (33), we examine the numerator and denominator terms. Letting $m_{\lambda^{(ex)}}^{(l)}$ be the expected value of each entry of $\lambda_l^{(ex)}$, we can write the numerator of the Q-function argument as

$$\begin{aligned} \mathbb{E}[\beta'_l] &= m_\lambda \left(1 + \sum_{i=1}^l \frac{1}{r^i \prod_{j=1}^i \bar{g}'_j} \right) \sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k \\ &+ \sum_{i=1}^l \frac{m_{\lambda^{(ex)}}^{(i)}}{r^i \prod_{j=1}^i \bar{g}'_j} \sum_{k=1}^b (\mathbf{w}_1^T \mathbf{B}_{ex})_k. \end{aligned} \quad (41)$$

Assumption 9. We will assume that the entries of λ and $\lambda_l^{(ex)}$ are statistically independent of each other at a given iteration l as is often done in density evolution studies. Further, we assume that the entries of $\lambda_l^{(ex)}$ are independent from iteration to iteration, as was implicitly assumed in [20].

Letting σ_l^2 be the variance of each entry of the vector $\lambda_l^{(ex)}$, we can write the squared-denominator of the Q-function argument as

$$\begin{aligned} \text{VAR}[\beta'_l] &= 2m_\lambda \left(1 + \sum_{i=1}^l \frac{1}{r^i \prod_{j=1}^i \bar{g}'_j} \right)^2 \sum_{k=1}^a (\mathbf{w}_1^T \mathbf{B})_k^2 \\ &+ \sum_{i=1}^l \frac{\sigma_i^2}{(r^i \prod_{j=1}^i \bar{g}'_j)^2} \sum_{k=1}^b (\mathbf{w}_1^T \mathbf{B}_{ex})_k^2. \end{aligned} \quad (42)$$

We are interested in the behavior of (41) as the number of iterations goes towards infinity. The divergence of the first series within (41) is not of interest as its effect will be canceled by the first series within (42) when they are combined into (33). We will apply the Ratio Test to the second series within (41), evaluating ρ as

$$\rho = \lim_{i \rightarrow \infty} \left| \frac{a_{i+1}}{a_i} \right| = \lim_{i \rightarrow \infty} \left| \frac{m_{\lambda^{(ex)}}^{(i+1)} r^i \prod_{j=1}^i \bar{g}'_j}{m_{\lambda^{(ex)}}^{(i)} r^{i+1} \prod_{j=1}^{i+1} \bar{g}'_j} \right|. \quad (43)$$

This series converges absolutely if $\rho < 1$ and diverges if $\rho > 1$. Noting that the gains \bar{g}'_l approach 1 rapidly with increasing l , this readily simplifies to

$$\rho = \lim_{i \rightarrow \infty} \left| \frac{m_{\lambda^{(ex)}}^{(i+1)}}{m_{\lambda^{(ex)}}^{(i)} r} \right|. \quad (44)$$

Thus, if the mean unsatisfied-check LLRs satisfy $m_{\lambda^{(ex)}}^{(i)} > Cr^i$ as $i \rightarrow \infty$, for some positive constant C , then $\rho > 1$, and (41) will grow without bound.

If we can be assured that (42) does not grow as fast as the square of (41), then the entire argument to the Q-function will grow without bound, driving the failure rate of the potential elementary trapping set towards zero. In that case, sufficient iterations and headroom for $\lambda_l^{(ex)}$ growth are the requirements for the model to achieve as low an error floor as desired.

Other models of error floor behavior [1], [20] have used the Gaussian approximation to LLR densities [40], which implies $\sigma_l^2 = 2m_{\lambda^{(ex)}}^{(l)}$. Moreover, using a numerical version of density evolution, it is found that $\sigma_l^2 < 2m_{\lambda^{(ex)}}^{(l)}$ as the LLRs get large

[43]. With an LLR variance of twice the mean or less, we find that the entire argument to the Q-function grows without bound in the cases that satisfy $m_{\lambda^{(ex)}}^{(i)} > Cr^i$. However, if the variance grows as the square of the mean, then the argument to the Q-function reaches a finite limit and a nonzero error floor is produced. This is a very important issue that we will revisit later in the paper.

E. Bounds on Spectral Radius

Using Perron-Frobenius theory, Schlegel and Zhang [20] derived an approximation to the spectral radius r of absorbing sets, which is

$$r \approx d_v - 1 - b/a. \quad (45)$$

We explore the accuracy of this approximation in Appendix C.

We now need some bounds on r , the spectral radius of the \mathbf{A} matrix, in order to compare it with the growth rate of $m_{\lambda^{(ex)}}^{(i)}$ which we will develop later. While the spectral analysis of digraphs is an active field of research, the classic bounds of Frobenius are sufficient for our needs.

Lemma 4. Let $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{A} \geq \mathbf{0}$ and let the i th row sum be $r_i = \sum_{j=1}^m [\mathbf{A}]_{ij}$. Then the maximal eigenvalue r is bounded above and below by

$$\min_{1 \leq i \leq m} r_i \leq r \leq \max_{1 \leq i \leq m} r_i. \quad (46)$$

If \mathbf{A} is irreducible, then the inequalities are strict unless all row sums are equal. The analogous statements will hold for the column sums of \mathbf{A} .

Proof: See pp. 8 and 22 of [30], and 492 of [27]. ■

Theorem 5. Consider a variable-regular LDPC code with variable-degree $d_v \geq 3$. Let trapping set \mathcal{S} induce an elementary connected subgraph $B_{\mathcal{S}}$ with $a \geq 2$ variable nodes and $b > 0$ degree-one check nodes. Further, let the associated undirected multigraph G be leafless. Then, the associated $(0, 1)$ -matrix $\mathbf{A}(D)$ must have spectral radius r such that $1 \leq r < d_v - 1$.

Proof: We have ruled out codewords by requiring $b > 0$. The codewords of a variable-regular graph will have an \mathbf{A} matrix with uniform row sums equal to $d_v - 1$, as each variable node in the subgraph is adjacent to d_v degree-two check nodes and the edge being updated is always excluded from computation in the SPA. By assumption, there will be some $b > 0$ degree-one check nodes. If we let b_{max} be the greatest number of degree-one check nodes neighboring any single variable node ($1 \leq b_{max} \leq d_v - 2$), and b_{min} be the least such number ($0 \leq b_{min} \leq b_{max}$), then using Lemma 4, we can state

$$d_v - 1 - b_{max} \leq r \leq d_v - 1 - b_{min}. \quad (47)$$

If \mathbf{A} is irreducible, then either no equalities above hold or both equalities hold, the latter when $b_{min} = b_{max}$. If \mathbf{A} is irreducible and $b_{min} = b_{max}$, then $1 \leq r \leq d_v - 2$. If \mathbf{A} is irreducible and $b_{min} < b_{max}$, then $1 < r < d_v - 1$. In the case that \mathbf{A} is reducible, Appendix A shows that the graphs

permitted by the connected and leafless conditions only allow $r = 1$, which will be strictly less than $d_v - 1$ since we have assumed $d_v \geq 3$. ■

Corollary 6. *If in addition to the conditions given in Theorem 5, we also stipulate that $b_{\min} < d_v - 2$, then the spectral radius r is bounded by $1 < r < d_v - 1$.*

Proof: The case where \mathbf{A} is irreducible and $b_{\min} = b_{\max}$, now yields $1 < r \leq d_v - 2$. The case where \mathbf{A} is irreducible and $b_{\min} < b_{\max}$ remains unchanged with $1 < r < d_v - 1$. Finally, the case where \mathbf{A} is reducible is not permitted, since Appendix A shows that all the reducible matrices that are associated with connected and leafless subgraphs have $d_v - 2$ degree-one check nodes neighboring every variable node. ■

V. NUMERICAL RESULTS WITH SATURATION

This section completes the prediction model by presenting the techniques used to produce the linear system's inputs and then compares predicted error floors to simulation results for three LDPC codes. Two numerical models are described for use as system inputs for the mean and variance of the unsatisfied, degree-one check nodes within the trapping set. These models also generate the modified mean gain \bar{g}_l of the degree-two check nodes required by the model presented in Section III. This completes the necessary tools to produce error floor predictions when the LLR metrics are saturated within the log-domain SPA decoder.

The first technique is the numerical version of density evolution known as discretized density evolution (DDE) described in [41]. This technique utilizes a probability mass function (pmf) that closely approximates the continuous probability density function (pdf) of node inputs and outputs. This technique is selected for the saturated case as it allows us to precisely model the behavior of LLRs as they saturate since it works directly on the pmfs. At a particular E_b/N_0 value, we capture the mean and variance of the values we require at each iteration for insertion into our model. See Table I for an example, which also lists the mean check node gain, which have been computed following (14).

The second technique is to simply run the log-domain SPA decoder with early termination turned off and collect the required statistics. Configuring the AWGN channel for a particular E_b/N_0 value, we capture the mean and variance of the check nodes' output LLRs and the mean check node gain computed as in (14) at each iteration. For an example of the results using this technique see Table II. We find one-hundred frames to be sufficient, and of course very quick to produce.

The results in the two tables are similar, as expected. Even though the unsatisfied check node variance is larger for the SPA simulation technique, we found the two techniques produced nearly identical error floor predictions for the codes presented. We chose to present the results obtained using the SPA simulation technique in the figures.

The first code we examine is the (640, 192) quasi-cyclic (QC) LDPC code from [14], with $d_v = 5$ and irregular check degree. The dominant trapping set for a saturating log-domain SPA is the (5, 5) elementary trapping set shown in [14]. The

TABLE I
DDE NUMERICAL RESULTS BY ITERATION FOR (3, 6)-REGULAR CODE AT AN E_b/N_0 OF 2.8 dB WITH LLR SATURATION SET TO ± 25 AT CHECK NODE OUTPUT. LLR PDFS DISCRETIZED TO PMFS WITH A RESOLUTION OF 50/2047. IN THIS CASE, $m_\lambda = 3.8109$.

l	$m_{\lambda(e_x)}^{(l)}$	σ_l^2	\bar{g}_l
1	0.669	1.47	0.3242
2	1.315	2.81	0.4898
3	2.08	4.24	0.6243
4	3.11	5.94	0.7472
5	4.66	8.05	0.8586
6	7.21	10.74	0.9446
7	11.66	14.22	0.9891
8	19.67	16.59	0.9995
9	24.91	0.47	1.0000
10	25.00	0.00	1.0000

TABLE II
SPA SIMULATION RESULTS BY ITERATION FOR THE (2640, 1320) MARGULIS CODE WHICH IS (3, 6)-REGULAR AT AN E_b/N_0 OF 2.8 dB WITH LLR SATURATION SET TO ± 25 AT CHECK NODE OUTPUT. IN THIS CASE, $m_\lambda = 3.8109$.

l	$m_{\lambda(e_x)}^{(l)}$	σ_l^2	\bar{g}_l
1	0.675	1.49	0.3245
2	1.324	2.88	0.4897
3	2.09	4.36	0.6219
4	3.14	6.19	0.7425
5	4.73	8.83	0.8505
6	7.37	13.65	0.9338
7	12.08	25.82	0.9793
8	19.13	33.94	0.9956
9	23.66	11.86	0.9995
10	24.83	1.31	1.0000

\mathbf{A} matrix for this trapping set is primitive and has maximum eigenvalue $r = 3$. The multiplicity of this trapping set in the code is just 64 and all such sets have disjoint sets of variable nodes. Figs. 6, 7 and 8 show FER results for simulation and error floor prediction for this code at two levels of LLR saturation. Our error floor prediction model shows very fast convergence, attributed to the large eigenvalue (*i.e.*, $r \gg 1$), so we stop our model at 14 iterations.

The FER simulation results shown in these figures are produced by our log-domain SPA decoder, running on a computer, implemented in double-precision floating-point. This SPA decoder is non-saturating, in that computations are organized such that saturation will not occur as shown in (9). We have not witnessed our non-saturating decoder fail on the (5, 5) trapping sets, which were reported to dominate in [14]. Next, saturation was intentionally introduced into our decoder at levels shown in the figures' legends. The intentional saturation was introduced at a point corresponding to the output of the complete check node update. We simulate the transmission of the all-zero codeword for simplicity and take any other decoding result to be decoding failure or frame error. The simulations were run for a maximum of 200 iterations.

Fig. 6 shows the error floors predicted by (34), but with all check node gains forced to unity. Fig. 7 shows the same conditions, but now with the mean gains of (14) applied to our model. Fig. 8 again shows the same conditions, but

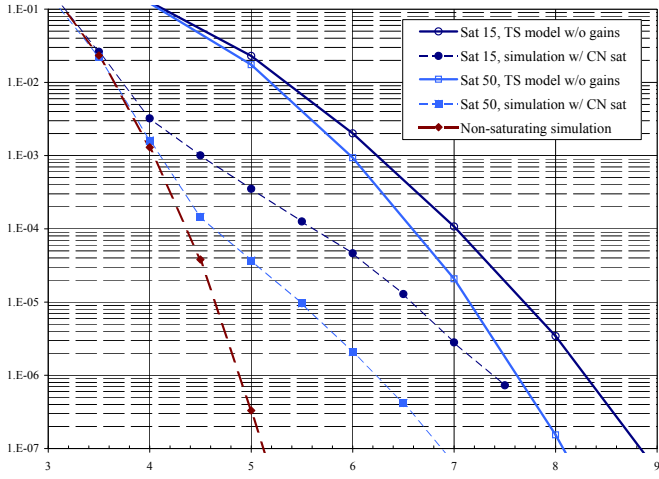


Fig. 6. FER vs. E_b/N_0 in dB for (640, 192) QC code with simulation and trapping set (TS) model predicted floor and using unit gains in model and no check node inversions.

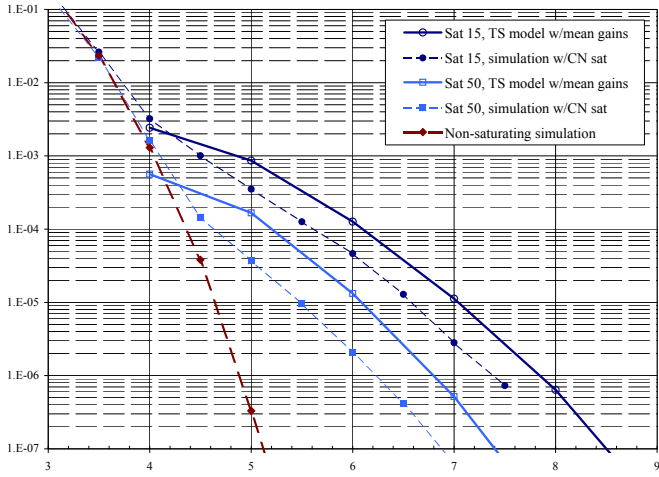


Fig. 7. FER vs. E_b/N_0 in dB for (640, 192) QC code with simulation and trapping set (TS) model predicted floor using mean gains in model and no check node inversions.

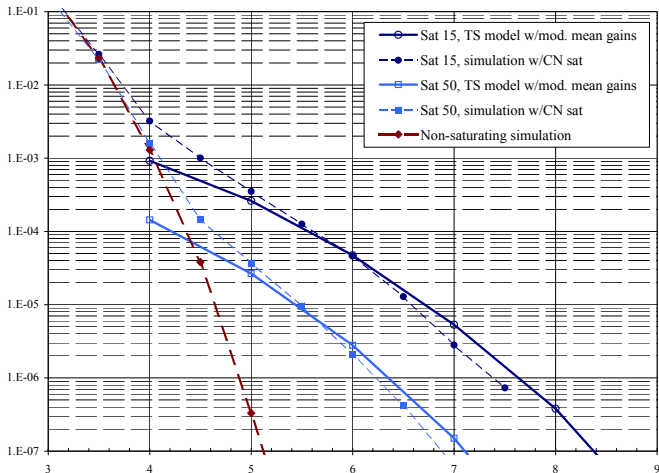


Fig. 8. FER vs. E_b/N_0 in dB for (640, 192) QC code with simulation and trapping set (TS) model predicted floor using modified mean gains in model.

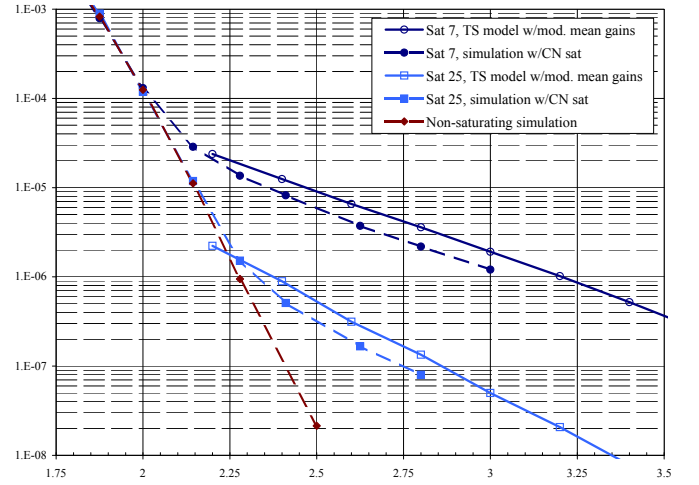


Fig. 10. FER vs. E_b/N_0 in dB for (2640, 1320) Margulis code with simulation and trapping set (TS) model predicted floor using modified mean gains.

this time with the modified mean gains of (18) applied to account for polarity inversions within the degree-two check nodes. We applied the inversion model for just the first 3 iterations. Comparing the first two figures, one notices that the mean gains provide a significant improvement in the model's accuracy, to within 0.5 dB of simulation. Comparing Figs. 7 and 8, one notices that the channel polarity model represents an improvement in the model's accuracy to within ± 0.1 dB of simulation.

Heuristically, the reason that error floor predictions drop when check node gains are introduced has to do with incorrect state buildup in early iterations. With unity gains, incorrect channel inputs may propagate rapidly in the highly interconnected states of the trapping set. With small gains in early iterations this is held off for several iterations, giving the unsatisfied check nodes a better chance to correct the system. As we previously noted, during early iterations polarity inversions through the degree-two check nodes act to effectively lower the check node gain. Therefore, as we'd expect, introducing the inversions lowers the predicted error floor.

The SPA results presented in [14] would appear slightly above the simulation curves with saturation set to 15 shown in Figs. 6 to 8. The FER ratio between our non-saturated SPA simulation and the SPA simulation of [14] is three orders of magnitude at an E_b/N_0 of 5 dB and growing very rapidly.

The second code analyzed is the (2640, 1320) Margulis LDPC code. It's a (3, 6)-regular code with dominant trapping sets reported to be the (12, 4) and (14, 4) elementary trapping sets, shown in Figs. 9a and 9b [7], [8], [14]. The A matrices for these two trapping sets are both imprimitive, with $h = 2$, and have maximum eigenvalues r of 1.6956 and 1.7606, respectively. We ran the model for 16 iterations. The multiplicity of each of these trapping sets is 1320 with lots of overlap between variable nodes of differing sets. In fact, each (14, 4) trapping set contains all 12 variable nodes and 2 of the unsatisfied check nodes of a (12, 4) trapping set, as can be seen in Fig. 9. Thus, when we compute the failure probability of one (12, 4) trapping set, it largely includes the

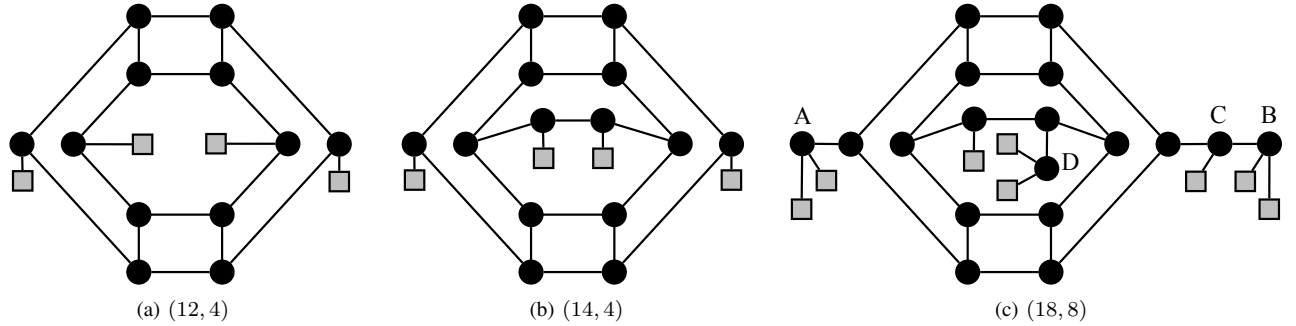


Fig. 9. Three elementary trapping sets of a $d_v = 3$ code shown with degree-one check nodes shaded and with degree-two check nodes omitted. The two on the left dominate the error floor of the saturated SPA decoder for the Margulis code, while the one on the right fails in the non-saturating SPA decoder.

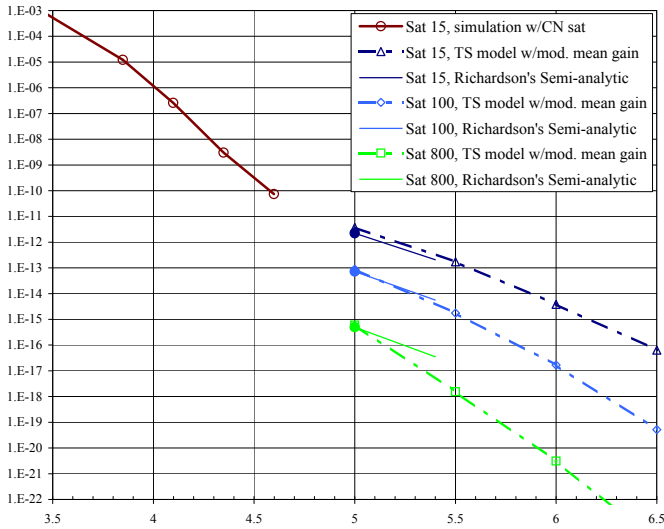


Fig. 11. BER vs. E_b/N_0 in dB for (2048, 1723) LDPC code from 802.3an with simulation, Richardson's semi-analytical (8, 8) floor estimation, and trapping set (TS) model predicted (8, 8) floor using modified mean gains.

failure probability of the (14, 4) trapping set that contains it. Therefore, Fig. 10 shows the error floors predicted for just the (12, 4) trapping sets, utilizing the model with modified mean gains. We applied the inversion model for just the first 5 iterations. We find that the state-space model overestimates the simulation's error floor by less than 0.2 dB.

The third code analyzed is the (2048, 1723) Reed-Solomon based LDPC code from the 802.3an standard. It's a (6, 32)-regular code with dominant trapping sets reported to be the (8, 8) elementary trapping set [20], [26]. The \mathbf{A} matrix for this trapping sets is primitive, with a maximum eigenvalue $r = 4$. The linear model for this trapping set converges very fast, in just 3 iterations when saturation is set to 15 and a few more iterations at higher saturation limits. We conservatively ran the model for 16 iterations. The multiplicity of this trapping sets is 14 272 [20]. Fig. 11 shows the predicted BER error floors of this code utilizing our model with modified mean gains if we assume there are no other trapping sets present in this code.

Since a standard Monte Carlo simulation of this low error floor takes a prohibitively large amount of CPU time, we have used Richardson's semi-analytical technique to estimate the error floor [8]. It is a form of importance sampling that

we describe in more detail in Section VII-B. We have run this semi-analytical technique at an E_b/N_0 of 5.0 dB (the solid circles on Fig. 11) and used Richardson's method of extrapolating the results to 5.4 dB (the solid lines). We have only configured the semi-analytical technique to measure the (8, 8) trapping sets. For the saturation levels of 15, 100, and 800, Richardson's technique required 0.9, 9.5, and 1300 hours of CPU time, respectively. We used less than 2 minutes of CPU time to collect the 1000-frames worth of LLR statistics that we used to produce each error floor prediction point. Our model's predictions appear to have an accuracy of better than 0.1 dB at 5.0 dB.

We chose these three LDPC codes as they are quite different and their error floors have appeared in prior studies. The trapping sets in these codes have a broad range of multiplicities and degree of overlap among them. Interestingly, the Margulis code has two dominant trapping sets, both with imprimitive \mathbf{A} matrices, while the other two codes are dominated by single trapping sets with primitive \mathbf{A} matrices. Regardless, we find that the state-space model with modified mean gains estimates the simulation's error floor reasonable well, to within 0.2 dB, and very quickly.

Overall, this is consistent with Schlegel and Zhang's results [20]. Their predicted error floor for the 802.3an code appears to be nearly on top of the simulated results of a hardware decoder. Our own floating-point simulations show that their hardware decoder has implementation losses, which we measure to be about 0.15 dB at the onset of the error floor.

Surprisingly, we find that LLR saturation at the fairly high values of 25, 50 and 100 produce noticeable error floors in simulation and model prediction. Finally, we note that the terms in (42) containing σ_i^2 have a small impact on the overall results.

VI. ANALYSIS USING DE WITHOUT SATURATION

The prior section addressed density evolution and predicted error floor levels when the LLR metrics within the decoder saturated. This section removes the saturation constraint to make ultimate error floor predictions. First, we must apply density evolution to model the contributions from nodes outside of the trapping set relevant to the behavior of our trapping set. Then we will refer back to the divergence condition of (44) to see if error floors are produced.

Assumption 10. Density evolution assumes that the Tanner graph has no cycles and the code's block length is infinite.

SPA decoding of LDPC codes exhibits a threshold phenomenon as the block length tends to infinity. The error rate drops very dramatically as the SNR exceeds the *decoding threshold* which can be found using density evolution [40]. As we are interested in the behavior of the error floor in this work, we assume that the channel SNR is always above the decoding threshold.

The application of this density evolution technique produces an analytical expression for the log-domain SPA LLRs. As density evolution progresses above the decoding threshold, LLR values grow fast, but the question is how fast. In [1], Sun showed that mean check node outputs, $m_{\lambda(ex)}$, grow from iteration $l-1$ to l as

$$m_{\lambda(ex)}^{(l)} = (d_v - 1)m_{\lambda(ex)}^{(l-1)} + \text{"some small value terms."} \quad (48)$$

We seek to bound the effect of Sun's "small value terms," by developing the governing expressions rather than the asymptotic result. We follow the development in [1], but we also develop bounds on the SNR region in which we can expect sufficient growth of $m_{\lambda(ex)}^{(l)}$ to eliminate error floors. Like [1], our focus is on the high E_b/N_0 regime.

We begin by using the consistent Gaussian approximation to AWGN density evolution [40]. The update rule at iteration l for the unsatisfied check nodes' output mean $m_{\lambda(ex)}$ is

$$m_{\lambda(ex)}^{(l)} = \phi^{-1} \left(1 - \left[1 - \phi \left(m_{\lambda} + (d_v - 1)m_{\lambda(ex)}^{(l-1)} \right) \right]^{d_c - 1} \right), \quad (49)$$

where $\phi(x)$ is defined by the integral

$$\phi(x) \triangleq \begin{cases} 1 - \frac{1}{\sqrt{4\pi x}} \int_{\mathbb{R}} \tanh \frac{u}{2} e^{-\frac{(u-x)^2}{4x}} du, & \text{if } x > 0 \\ 1, & \text{if } x = 0. \end{cases} \quad (50)$$

Note that $\phi(x)$ is continuous and monotonically decreasing from 1 to 0, as x increases from 0 towards ∞ .

The following lower and upper bounds on $\phi(x)$ are true for any $x > 0$ and tighten as x increases [40]:

$$\sqrt{\frac{\pi}{x}} e^{-x/4} \left(1 - \frac{3}{x} \right) < \phi(x) < \sqrt{\frac{\pi}{x}} e^{-x/4} \left(1 - \frac{1}{7x} \right). \quad (51)$$

We will relax the upper bound, for $x > 0$, to be

$$\phi(x) < \sqrt{\frac{\pi}{x}} e^{-x/4}. \quad (52)$$

We will take the lower bound of (51) and use it to upper bound $1 - \phi(x)$ as

$$1 - \delta \sqrt{\frac{\pi}{x}} e^{-x/4} > 1 - \phi(x), \quad (53)$$

where $\delta \triangleq 1 - 3/x$, noting that δ approaches 1 for large x . We take a small digression before proceeding with the main theorem to develop inequalities that we will require.

Lemma 7. For any $K \in \mathbb{R}$ and $x, \beta > 0$ there exists a unique $k_1 \in \mathbb{R}$ such that $K = k_1 + \beta \ln(1 + k_1/x)$. Moreover, $k_1 > -x$.

Proof: The function $f(k_1) = k_1 + \beta \ln(1 + k_1/x)$ is well-defined only on the domain $k_1 > -x$. It is continuous and monotonically increasing on that domain, and $f(k_1) \rightarrow -\infty$ as $k_1 \rightarrow -x$ from the right. ■

Lemma 8. Let $m + \beta \ln m > x + \beta \ln x + K$ where $m, x, \beta, K \in \mathbb{R}$ with $m, x, \beta > 0$. Then $m > x + \frac{K}{1+\beta/x}$.

Proof: First we will prove by contrapositive that $m > x + k_1$, where $K = k_1 + \beta \ln(1 + k_1/x)$. From Lemma 7 we know for any K we can find a unique real $k_1 > -x$.

Assume that $m \leq x + k_1$. Then we can write $\beta \ln m \leq \beta \ln(x + k_1) = \beta \ln x + \beta \ln(1 + k_1/x)$. Adding these two inequalities together yields

$$m + \beta \ln m \leq x + \beta \ln x + k_1 + \beta \ln(1 + k_1/x)$$

Recognizing that the right two terms combine to equal K , we have an expression that exactly contradicts our recent assumption. Thus, $m > x + k_1$.

Since we know for any $z > -1$ that $z \geq \ln(1 + z)$, we can write

$$K = k_1 + \beta \ln(1 + k_1/x) \leq k_1 + \beta k_1/x$$

This implies $k_1 \geq \frac{K}{1+\beta/x}$. Finally, we combine these two inequalities to produce the desired result.

$$m > x + k_1 \geq x + \frac{K}{1 + \beta/x}$$

Theorem 9. Given a (d_v, d_c) -regular LDPC code, with $d_v \geq 3$, at sufficiently high E_b/N_0 , after a finite number of iterations, the log-domain SPA decoder's check node output mean $m_{\lambda(ex)}^{(l)}$ satisfies $m_{\lambda(ex)}^{(l)} > C(d_v - 1)^l$.

Proof: We start by restating (49) in the form

$$1 - \phi \left(m_{\lambda(ex)}^{(l)} \right) = \left[1 - \phi \left(m_{\lambda} + (d_v - 1)m_{\lambda(ex)}^{(l-1)} \right) \right]^{d_c - 1}. \quad (54)$$

First, we'd like to apply the upper bound of (53) to the LHS of (54), where $x = m_{\lambda(ex)}^{(l)}$. To do this we must assume that $m_{\lambda(ex)}^{(l)} > 3$ so that $\delta > 0$. It will take some number of iterations for δ to get close to 1, depending upon the AWGN channel's E_b/N_0 , but when the SNR exceeds the decoding threshold, we are assured that the LLRs will continue growing by density evolution. Now,

$$1 - \delta \sqrt{\frac{\pi}{m_{\lambda(ex)}^{(l)}}} e^{-m_{\lambda(ex)}^{(l)}/4} > \left[1 - \phi \left(m_{\lambda} + (d_v - 1)m_{\lambda(ex)}^{(l-1)} \right) \right]^{d_c - 1}, \quad (55)$$

where $\delta = 1 - 3/m_{\lambda(ex)}^{(l)}$.

Next, we can truncate the binomial expansion to a simple bound, in the form shown below for small $\phi(x)$. Specifically, for any positive integer n ,

$$(1 - a)^n = 1 - na + \binom{n}{2}a^2 - \binom{n}{3}a^3 \dots + (-a)^n > 1 - na, \quad \text{for small enough } a \quad (56)$$

For example, by the time the LLRs have reached $m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(i-1)} > 8$, we are assured that $\phi(\cdot) < 0.09$ by (51), which is small enough for the truncated binomial (56) to hold as a positive lower bound for $d_c \leq 12$. Codes with higher check degree will require larger LLRs for (56) to hold. Utilizing this binomial bound we now have

$$\sqrt{\frac{\pi}{m_{\lambda(ex)}^{(l)}}} e^{-m_{\lambda(ex)}^{(l)}/4} < \frac{d_c - 1}{\delta} \phi\left(m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}\right). \quad (57)$$

Next, we apply the upper bound of (52) with $x = m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}$ to the right hand side of (57), take the logarithm of both sides, and scale by -4 to get

$$\begin{aligned} 2 \ln m_{\lambda(ex)}^{(l)} + m_{\lambda(ex)}^{(l)} &> -4 \ln \frac{d_c - 1}{\delta} \\ &+ 2 \ln \left(m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}\right) \\ &+ m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}. \end{aligned} \quad (58)$$

Applying Lemma 8 with $\beta = 2$ to (58) yields

$$m_{\lambda(ex)}^{(l)} > m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)} - \frac{4 \ln \frac{d_c - 1}{\delta}}{1 + \frac{2}{m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}}}. \quad (59)$$

Examining (59), we see that if the m_λ term exceeds the magnitude of the final term which is negative, we can guarantee $m_{\lambda(ex)}^{(l)}$ growth by at least a factor of $d_v - 1$ at each iteration. Noting that the mean channel LLR is $m_\lambda = 4RE_b/N_0$ in the AWGN channel, the condition needed to satisfy $m_{\lambda(ex)}^{(l)} > C(d_v - 1)^l$ is

$$R \frac{E_b}{N_0} > \left(\ln \frac{d_c - 1}{\delta} \right) \left[1 + \frac{2}{x} \right]^{-1}, \quad (60)$$

where $x = m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)} \gg 0$, and x is increasing with each iteration. Since $1 + 2/x > 1$ is always true, we can make the condition (60) simpler, but stronger as

$$R \frac{E_b}{N_0} > \ln \frac{d_c - 1}{\delta}, \quad (61)$$

where $\delta = 1 - 3/m_{\lambda(ex)}^{(l)} > 0$. The value of δ approaches 1 as the number of iterations increases. If our earlier assumptions on the magnitude of the LLRs and (61) are satisfied, we have reached the LLR growth regime of $m_{\lambda(ex)}^{(l)} > (d_v - 1)m_{\lambda(ex)}^{(l-1)}$. ■

Example 6. For any $(3, 6)$ -regular code, $R = 0.5$, we compute the E_b/N_0 threshold of (61) to be 5.077 dB, assuming that $\delta \approx 1$. At any E_b/N_0 greater than 5.077 dB, we can guarantee that $m_{\lambda(ex)}^{(i)} > C(d_v - 1)^i$ is reached after enough iterations to satisfy the assumptions made in this section. This is a sufficient growth rate to overcome any potential elementary trapping set, because $r < d_v - 1$ as proved in Theorem 5.

Corollary 10. Given a (d_v, d_c) -regular LDPC code, with $d_v \geq 3$, at any E_b/N_0 above the decoding threshold, after a finite number of iterations l , the log-domain SPA decoder's check node output mean $m_{\lambda(ex)}$ satisfies $m_{\lambda(ex)}^{(l)} > Cr^l$, for any r , such that $1 \leq r < d_v - 1$.

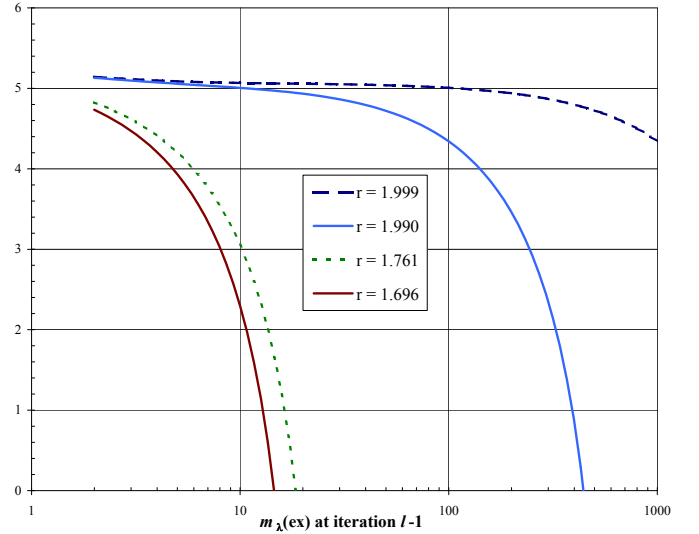


Fig. 12. Threshold E_b/N_0 in dB vs. $m_{\lambda(ex)}^{(l-1)}$ to achieve LLR growth needed to overcome a trapping set characterized by r as specified in the legend for $(3, 6)$ -regular codes such as Margulis.

Proof: The condition we want to prove is

$$\frac{m_{\lambda(ex)}^{(l)}}{m_{\lambda(ex)}^{(l-1)}} > r, \quad (62)$$

which, by (59) of the previous proof, will be true if the following stronger condition holds

$$d_v - 1 + \frac{m_\lambda}{m_{\lambda(ex)}^{(l-1)}} - \frac{1}{m_{\lambda(ex)}^{(l-1)}} \frac{4 \ln \frac{d_c - 1}{\delta}}{1 + \frac{2}{x}} > r, \quad (63)$$

where $x = m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}$. Define ϵ such that $\epsilon \triangleq d_v - 1 - r > 0$. Rearranging terms, we have the equivalent condition

$$R \frac{E_b}{N_0} > \left(\ln \frac{d_c - 1}{\delta} \right) \left[1 + \frac{2}{x} \right]^{-1} - \frac{\epsilon m_{\lambda(ex)}^{(l-1)}}{4}, \quad (64)$$

which will be true for some l given any positive growth in $m_{\lambda(ex)}^{(l)}$. ■

The most difficult elementary trapping sets to overcome, in our context, have a spectral radius very close to $d_v - 1$, which is the upper bound on spectral radii according to Theorem 5. To approach such an extreme spectral radius, the trapping set would require a very large a and small b parameter as suggested by (45). (See Appendix C for the range of measured spectral radii of a large number of trapping sets.) The E_b/N_0 required to reach the LLR growth regime of $m_{\lambda(ex)}^{(i)} > Cr^i$, which is needed to overcome the trapping behavior of a specific trapping set, may be significantly less than the threshold of (61) as the next example illustrates.

Example 7. Continuing the prior example, we note that the $(2640, 1320)$ Margulis code is $(3, 6)$ -regular, with the most troublesome trapping sets reportedly being $(12, 4)$ and $(14, 4)$ [7], [8], [14]. We have computed their maximum eigenvalues r to be 1.6956 and 1.7606, respectively. Fig. 12 shows the behavior of the threshold pair $(E_b/N_0, m_{\lambda(ex)}^{(l-1)})$,

that is required by (64) to achieve a growth rate sufficient to overcome those specific eigenvalues and two hypothetical extreme eigenvalues of 1.999 and 1.99. From Fig. 12 we see that for the (12, 4) trapping set at an E_b/N_0 of 2.8 dB we will require $m_{\lambda(ex)}^{(l-1)} > 8.6$ to enter the LLR growth regime of $m_{\lambda(ex)}^{(l)} > C(1.6956)^i$ necessary to eventually overcome that trapping set. As can be seen in Table I, this growth rate condition is satisfied at the start of the eighth iteration.

Theorem 11. *Given a variable-regular, but check-irregular $(d_v, \rho(x))$ LDPC code, with $d_v \geq 3$, at sufficiently high E_b/N_0 , after a finite number of iterations, the log-domain SPA decoder's check node output mean $m_{\lambda(ex)}^{(l)}$ satisfies $m_{\lambda(ex)}^{(l)} > C(d_v - 1)^l$.*

Proof: Let d_r denote the maximum check degree for the graph and let $\rho(x)$ represent a polynomial with coefficients $\rho_j, j \in \{2, \dots, d_r\}$, denoting the fraction of edges belonging to degree- j check nodes. If we separate $m_{\lambda(ex)}^{(l)}$ into its components we can reuse our results from Theorem 9. Given

$$m_{\lambda(ex)}^{(l)} = \sum_{j=2}^{d_r} \rho_j m_{j,\lambda(ex)}^{(l)}, \quad (65)$$

where

$$m_{j,\lambda(ex)}^{(l)} = \phi^{-1} \left(1 - [1 - \phi(x)]^{j-1} \right), \quad (66)$$

$$x = m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}, \quad (67)$$

and letting $\delta_j \triangleq 1 - 3/m_{j,\lambda(ex)}^{(l)}$, (59) implies

$$m_{\lambda(ex)}^{(l)} > m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)} - \sum_{j=2}^{d_r} \rho_j \frac{4 \ln \frac{j-1}{\delta_j}}{1 + \frac{2}{m_\lambda + (d_v - 1)m_{\lambda(ex)}^{(l-1)}}}. \quad (68)$$

As before, the growth condition is always met with an even stronger condition on the SNR, stated as

$$R \frac{E_b}{N_0} > \sum_{j=2}^{d_r} \rho_j \ln \frac{j-1}{\delta_j}. \quad (69)$$

When (69) is satisfied, then $m_{\lambda(ex)}^{(l)} > (d_v - 1)m_{\lambda(ex)}^{(l-1)}$. ■

Corollary 12. *Given a variable-regular, but check irregular $(d_v, \rho(x))$ LDPC code, with $d_v \geq 3$, at any E_b/N_0 above the decoding threshold, after a finite number of iterations l , the log-domain SPA decoder's check node output mean $m_{\lambda(ex)}^{(l)}$ satisfies $m_{\lambda(ex)}^{(l)} > Cr^l$, for any r , such that $1 \leq r < d_v - 1$.*

Proof: Similar to Corollary 10. ■

When combined with (44) developed in Section IV, this section has shown that every potential (non-codeword) elementary trapping set error can be corrected by a non-saturating log-domain SPA decoder after a sufficient number of iterations, provided we can rely upon density evolution as a model for LLR growth outside of the trapping set for variable-regular ($d_v \geq 3$) LDPC codes.

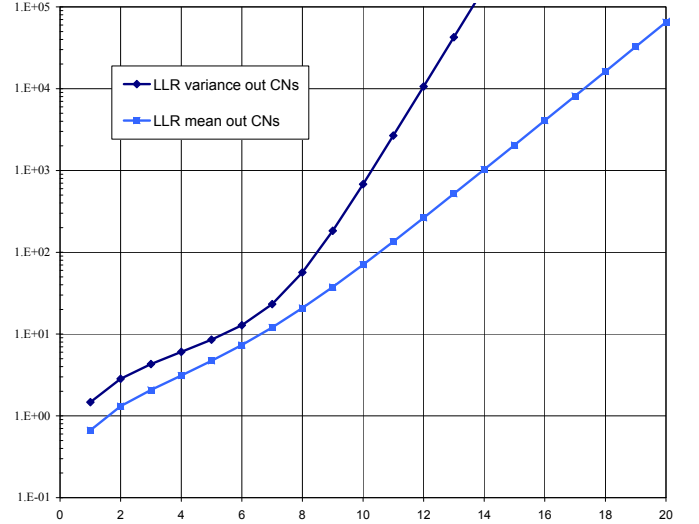


Fig. 13. Check node output LLR mean and variance versus iteration number for Margulis Code at E_b/N_0 of 2.8 dB.

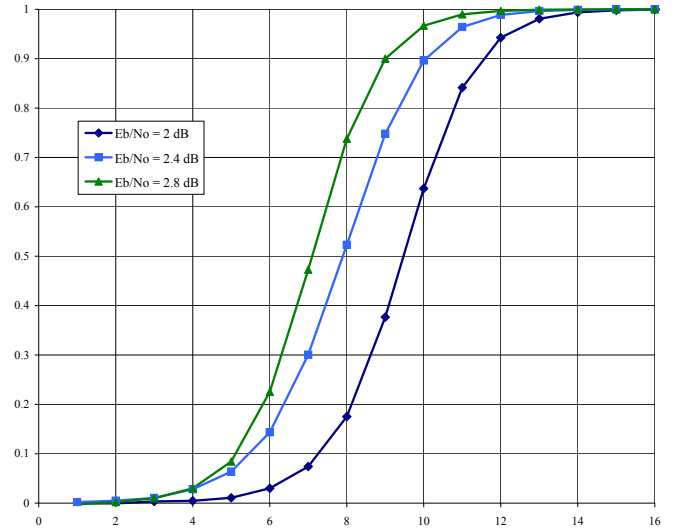


Fig. 14. Correlation coefficient among the four LLR check node output metrics used as linear model inputs versus iteration number for (12, 4) trapping sets of the Margulis code as measured in log-domain SPA simulation.

VII. MARGULIS RESULTS WITHOUT SATURATION

A. Applying the Error Floor Prediction Model

The previous section drew conclusions on non-saturating log-domain SPA decoder performance using density evolution techniques. Density evolution (DE) assumes that the Tanner graph does not contain cycles as it takes the statistical distributions to be independent among incoming edges. Useful finite-length LDPC codes *do* contain cycles, so it's worth questioning the use of DE to predict error floors.

In Fig. 13 we plot the mean and variance of check node output LLRs from the log-domain SPA decoder simulation of the (2640, 1320) Margulis code. We have noted that the mean follows very closely the mean LLR predicted by density evolution. This mean also follows very closely the lower bounds we developed in the previous section once the LLR

is greater than ten. The variance of the LLRs from the check nodes, however, shows a problem. For the first seven iterations, the variance is twice the mean as predicted by the Gaussian approximation to DE. However, by the ninth iteration, the variance has clearly taken on the trend of the square of the mean. As discussed in Section IV, this trend can generate an error floor as the mean to standard deviation ratio of β'_l in (33) reaches a fixed value rather than growing as iterations get large, *i.e.*,

$$\frac{\mathbb{E}[\beta'_l]}{\sqrt{\text{VAR}[\beta'_l]}} = O_l(1). \quad (70)$$

The cause of the variance's departure from the behavior predicted by DE is the positively correlated nature of the LLRs after several iterations. In Fig. 14 we plot the average of the off-diagonal values from the 4×4 matrix of correlation coefficients of the four LLR check node output metrics used as linear model inputs versus the iteration number for the (12, 4) trapping sets of the Margulis code. This shows significant positive correlation appearing in the sixth iteration and progressing rapidly. Thus, the assumptions used to justify (42) don't hold in this case. The (640, 192) QC code studied also shows similar variance and correlation behavior with an earlier onset, due to its smaller girth (6 vs. 8).

Also exposing flaws in our probability of error model is the skewness of β'_l . We have measured its skewness in the range of 0.6 to 1.8 as the correlated LLRs propagate through a log-SPA simulation of the Margulis code. Skewness is a measure of the asymmetry of the probability distribution of a random variable. Of course, (33) assumed a Gaussian distribution, which is symmetric about the mean and has a skewness of zero.

The problems presented here make it impossible to use the probability model that we presented with any accuracy beyond the sixth iteration of decoding the Margulis code without saturation.

B. Modifying and Applying Richardson's Semi-analytical Technique

First we introduce Richardson's semi-analytical technique to estimate the error floor by simulating in the vicinity of trapping sets [8]. Then, we will describe the changes we need to the technique for it to be effective with a non-saturating decoder, and present our results.

Consider a specific, known trapping set \mathcal{T} containing a variable nodes. We will generate the noise for the variables outside the trapping set as we normally would for standard Monte-Carlo simulation, but we will treat the noise within the trapping set in a special way. Consider that the noise subvector $\mathbf{n} = (n_1, \dots, n_a)$ containing i.i.d. zero-mean Gaussian random samples may be generated in various domains. If the elements of \mathbf{u} are i.i.d. Gaussian then so are the elements of $\mathbf{n} = \mathbf{Q}\mathbf{u}$, if \mathbf{Q} is orthogonal. Multiplying by an orthogonal matrix \mathbf{Q} preserves the i.i.d. nature as $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and preserves the Gaussian distribution as sums of Gaussians are also Gaussian-distributed.

Richardson effectively introduces a special orthogonal matrix \mathbf{Q} with a column that has equal support in the a positions

of the known trapping set. The remainder of the orthogonal matrix may be filled using Gram-Schmidt. Consider the first column of \mathbf{Q} to be $[1/\sqrt{a}, \dots, 1/\sqrt{a}]^T$ and the first element of \mathbf{u} to be $s\sqrt{a}$, where s is $N(0, \sigma^2/a)$. This allows us to treat the Gaussian random variable s separately as an analytical function, while we perform Monte-Carlo on the other $a - 1$ dimensions of the noise subvectors with variance σ^2 .

Now, we fix s at several negative values and run simulations, finding the conditional trapping set failure probability $P\{\xi_{\mathcal{T}}|s\}$ for each s value. Then, the overall trapping set failure probability $P\{\xi_{\mathcal{T}}\}$ may be computed using total probability, as

$$P\{\xi_{\mathcal{T}}\} = \int P\{\xi_{\mathcal{T}}|s\}p_S(s)ds \quad (71)$$

Due to the high degree of overlap among trapping sets, when counting failures for $P\{\xi_{\mathcal{T}}|s\}$, Richardson only counts a frame as a failure if the set of symbols which are not eventually correct exactly matches the symbol locations of \mathcal{T} , the specific trapping set under test.

In the case of the (14, 4) trapping set of the Margulis code, the not-eventually-correct symbols exactly match the symbols in \mathcal{T} , when testing the saturated decoder at a rate of about 3 in 4 failures or greater (at $s = -1.35$). Note that this rate drops substantially as s approaches -1 . However, for a non-saturating decoder, this ratio became less than 1 in 10^5 (at $s = -1.35$) in our tests, indicating that the not-eventually-correct condition is not effective when saturation is eliminated. This ratio was so low when running Richardson's technique for \mathcal{T} set to a (12, 4) trapping set without saturation, that we couldn't measure it. When saturation is present, failures are generally forced to occur on absorbing sets, and this is often the set under test when $s \leq -1.25$. Without saturation present, the failures occur on a wide variety of Tanner subgraphs, most larger than the original a variables under test. The decoder failures appear to be structures such as the (18, 8) subgraph shown in Fig. 9c and larger, which are generally not absorbing sets. The set of general Tanner subgraphs within the Margulis code is vast [11]. Rerunning the error floor estimate for each type of subgraph with such a low capture rate would be impractical. We propose two solutions to this.

The first solution would be to ignore the not-eventually-correct criteria and simply count all failures at the risk of overestimating the error floor. Instead, we propose to estimate the error floor by introducing saturation in the final iterations of failed frames so that they may fail on absorbing sets. These final iterations with saturation force corrections to labeled variable nodes of Fig. 9c, as the unsatisfied check nodes outnumber the satisfied checks, driving the failed structure to the (14, 4) absorbing set contained within. We prefer this approach as it reduces the number of incorrect bits per failure and is perhaps a useful strategy in practice.

As there may be many variable nodes to correct to get to the absorbing set contained within the failed structure we allow for 20 iterations of saturated SPA decoding before beginning the "eventually correct" logic which runs for 12 additional iterations. We chose to run the saturation phase at an LLR limit of 25. Our results are shown in Fig. 15 where

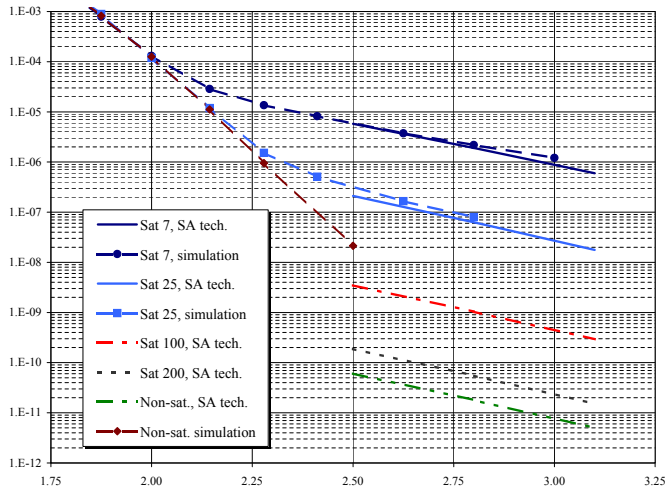


Fig. 15. FER vs. E_b/N_0 in dB for Margulis LDPC Code in AWGN. The semi-analytical (SA) technique counts only the (12,4) and (14,4) trapping sets.

all simulations were run for a maximum of 200 iterations, except that the non-saturating semi-analytical simulations were run for 32 additional iterations as just described. The semi-analytical technique was run at $E_b/N_0 = 2.8$ dB in all cases, and was extrapolated locally as suggested in [8]. Richardson's extrapolation assumes that the conditional failure probability $P\{\xi_{\mathcal{T}}|s\}$ is relatively insensitive to SNR changes. Fig. 15 shows an estimated error floor at $E_b/N_0 = 2.8$ dB of 2×10^{-11} FER due to these two trapping sets, which is incrementally better than the 6×10^{-11} achieved by limiting the LLRs to 200 at the check node output. In making this error floor prediction we assume that no other trapping sets become dominant in these conditions.

Several additional observations are worth noting. Increasing the maximum number of iterations of the non-saturating simulation to 1000 reduces the error floor by about a factor of 3. Additionally, we have applied the semi-analytical techniques of this section to estimate the FER contribution of the (12,4) and (14,4) trapping sets at 4×10^{-17} FER at $E_b/N_0 = 4$ dB. This shows that the error floor falls significantly faster than Richardson's extrapolation predicts, since the conditional failure probability $P\{\xi_{\mathcal{T}}|s\}$ appears to decrease significantly with increasing SNR. As shown in Section VI, at higher SNR, the beneficial LLR growth starts earlier and grows faster giving the code beyond the trapping set a higher probability to correct the channel error along a trapping set for a non-saturating decoder. However, an SPA decoder with limited LLR dynamic range will not leverage these effects that depend on SNR.

Table III shows the ratio of the error floor contributions of the (12,4) trapping set to the (14,4) at $E_b/N_0 = 2.8$ dB. We note that when saturation limits are low the (12,4) trapping set dominates the error floor, but when the limits are raised (14,4) dominates. There are two potential reasons for this. The first is that the larger maximum eigenvalue value of (14,4) is more sensitive to saturation even though it's less likely to initially trigger due to the greater number of variable nodes involved. Secondly, an effect we have noticed is that a growing fraction of (12,4) trapping failures are subsumed by

TABLE III
RATIO OF (12,4) TRAPPING SET ERROR FLOOR CONTRIBUTION TO (14,4) FOR THE MARGULIS CODE AT $E_b/N_0 = 2.8$ dB USING RICHARDSON'S SEMI-ANALYTICAL TECHNIQUE.

Decoder	Ratio
Saturated SPA, at ± 7 CN out	4.6 : 1
Richardson's 5-bit [8]	3.3 : 1
Saturated SPA, at ± 15 CN out	3.3 : 1
Saturated SPA, at ± 25 CN out	1.7 : 1
Saturated SPA, at ± 50 CN out	0.7 : 1
Saturated SPA, at ± 100 CN out	0.6 : 1
Saturated SPA, at ± 200 CN out	0.46 : 1
Non-saturating SPA	0.4 : 1

the (14,4) absorbing sets which contain them as LLR limits are raised. Additionally, we notice that the (14,4) trapping sets dominate even more as the SNR is increased for a non-saturating decoder.

Finally, we would like to know if we are seeing a "floor" or not for the non-saturating SPA results. Extrapolating the waterfall region of Fig. 15 at a constant log-log slope out to $E_b/N_0 = 4.0$ dB yields an FER of about 1×10^{-19} . Thus, the FER contribution of the (12,4) and (14,4) trapping sets at 4×10^{-17} FER does overpower the extrapolated waterfall FER at 4.0 dB. However, early evidence suggests that the effect is about twice as steep versus SNR than what was previously referred to as a "floor" for the Margulis code. Thus, our floor results are about 5 orders of magnitude lower at $E_b/N_0 = 2.8$ dB than prior SPA results, and about 8 orders of magnitude lower at 4.0 dB.

VIII. CONCLUSION

We have drawn into question the error floor levels published in [7], [8], [14] and their root cause. We have shown that the error floor levels are caused by the interaction of non-codeword trapping sets and the numerical effects of handling highly-certain messages in a belief propagation decoder, such as the log-domain SPA. We have shown that when care is taken in processing those messages the error floor level may be lowered by many orders of magnitude.

We have added some clarity to the situation of two distinctly different applications of the linear system model. On one hand, Sun introduced the state-space model and proved that many error floors don't exist for infinite length codes without cycles [1]. On the other hand, Schlegel and Zhang improved upon the model and predicted nonzero error floors due to the dominant (8,8) trapping sets of the 802.3an code [20].

With respect to Sun's work, we are in agreement with Sun's conclusion as applied to variable-regular LDPC codes without cycles. While Sun's analysis was asymptotic, ours used non-asymptotic bounds to derive the SNR thresholds and SNR-LLR regions in which the beneficial metrics from the degree-one check nodes grow fast enough to overcome the potentially faulty channel information that triggers a trapping set failure. Further, we have shown empirically that the assumptions in the probability of error model do not hold for codes with cycles. For these codes, we showed that the effects of positively correlated LLRs when allowed to grow (without saturation)

drive the ratio of the mean to standard deviation to a finite value possibly implying an error floor. This error floor level is difficult to estimate due to the positively correlated LLRs and the substantially non-Gaussian distribution of our error indicator.

We are also able to put Schlegel and Zhang's work into perspective. While not stated outright, their error rate results showing a nonzero error floor for a trapping set were for a decoder that saturated (*i.e.*, "clipped") the LLR values. They made no claims with regard to the effects of removing saturation in [20]. Their addition of mean gains to the model makes a substantial improvement to error floor prediction as we have shown. Our model is more general than theirs, in that we do not require that $\mathbf{B} = \mathbf{B}_{ex}$, that \mathbf{B} has regular column weight, and that \mathbf{A} has regular row and column weight. Additionally, we have simplified considerably the incorporation of polarity inversion by the degree-two check nodes into the model. While Schlegel and Zhang mention that general density evolution is used to drive their model, we have described using either discretized density evolution or a rapid 100-frame simulation to collect the statistics needed to drive the model. We have shown that the reason the model predicts saturated performance relatively well is that by the iteration count at which LLRs become substantially correlated, their variance is driven to zero by the effect of saturation. This means that the model derived for the case with no cycles actually works rather well for the case with cycles when there is a significant saturation effect present. In our experiments the model with modified mean gains estimated the FER floor of floating-point simulation to within 0.2 dB for three different LDPC codes.

Altogether, we have reached a better understanding of trapping set failure mechanisms. Trapping set failures can be frequently corrected when the beneficial unsatisfied check LLRs eventually overpower the trapping set's own detrimental LLR growth. Achieving this reliably can take significant LLR metric dynamic range. The greater the LLRs are allowed to grow the more often channel inversions aligned to trapping sets can be overcome. Chen et al. had the insightful observation in [23], "...that the error floor is caused by the combined effects of short cycles in the graph representation of the code and of clipping."

This phenomenon also helps explain prior observations. Others have noted that "stable" trapping set failures converge very rapidly (*cf.* §15.3.1 of [44]), often in less than ten iterations. We now realize that this is due to the saturation point of the beneficial unsatisfied check nodes being reached and bringing the correction process to a halt. When it's reached and the states are still in error the structure is very stable when it's an absorbing set. Absorbing sets guarantee that unsatisfied check messages are always outnumbered by internal messages at any variable node within the set. Metric saturation is why the bit-flipping decoder analogy was made in [13] to explain absorbing sets. The process of passing real-valued messages in a log-domain SPA degenerates to passing positively or negative saturated messages where the outcomes can be explained by bit-flipping decoder logic. We have shown that when saturation degenerates the SPA to a bit flipping decoder, it actually

helps to correct variable nodes in the trapping set until an absorbing set is reached. We have used this observation to modify Richardson's semi-analytical technique so that it may estimate error floors of a non-saturating SPA decoder.

The results make clear that when presenting error floors, future researchers should document the levels of any LLR saturation.

Future work may include extensions to a larger collection of trapping sets, to variable-irregular graphs, and to more channels. We are also interested in improvements to the model, such as accounting for correlated LLRs.

APPENDIX A REDUCIBLE \mathbf{A} MATRICES

In this Appendix we address the special case of reducible \mathbf{A} matrices.

Theorem 13. *Consider a connected multigraph G which meets Assumption 7. G is either a cycle graph or has an associated $\mathbf{A}(D)$ matrix that is irreducible.*

Proof: Multigraphs with any vertices of degree one are not allowed by Assumption 7. Connected multigraphs with all vertices of degree two are cyclic by Lemma 1. So now we'll show any possible remaining connected multigraphs must have an irreducible adjacency matrix $\mathbf{A}(D)$.

The remaining multigraphs must have at least one vertex of degree greater than two. Also, all vertex degrees must sum to an even number, as every edge joins two vertices. This yields a connected multigraph with at least two cycles. Since a walk that does not backtrack through a connected multigraph with two cycles is able to reverse direction, every edge may be visited in either direction. Thus, when the edges of G are expanded to be vertices of digraph D , D will be strongly connected. Since D is strongly connected, its associated adjacency matrix $\mathbf{A}(D)$ will be irreducible by Lemma 2. ■

Of course, since $\mathbf{A}(D)$ is irreducible so must be $\mathbf{A}(D)^T$, which we use as \mathbf{A} in our state-space model.

Therefore, the remainder of this section need only address the properties of cycle graphs. The expansion of the edges of cycle graph $G = C_a$ to digraph D will produce two disconnected directed cycles of length a , one associated with the clockwise non-backtracking walk of G and one with the counter-clockwise non-backtracking walk of G . Thus, $\mathbf{A}(D)$ may be symmetrically permuted to the following form, in which both \mathbf{A}_1 and \mathbf{A}_2 are irreducible.

$$\mathbf{P}\mathbf{A}(D)\mathbf{P}^T = \begin{pmatrix} \mathbf{A}_1 & 0 \\ 0 & \mathbf{A}_2 \end{pmatrix} \quad (72)$$

Since the component digraphs are each directed cycles of length a , for some P , both adjacency submatrices \mathbf{A}_1 and \mathbf{A}_2 are $a \times a$ permutation matrices that implement a circular shift by one position. Thus, both \mathbf{A}_1 and \mathbf{A}_2 are imprimitive with $h = a$. Since every row and column of \mathbf{A}_1 and \mathbf{A}_2 are weight one, all their eigenvalues lie on the unit circle, $r = \rho(\mathbf{A}_1) = \rho(\mathbf{A}_2) = 1$. The eigenvectors of \mathbf{A}_1 and \mathbf{A}_2 must be identical. The eigenvectors of \mathbf{A}_1 are turned into the eigenvectors of \mathbf{A} by appending a zeros. As \mathbf{A}_1 is a permutation matrix, the eigenvalue $r = 1$ is associated with the all-ones eigenvector

(or a scalar multiple thereof): $\mathbf{A}_1 \mathbf{v}_{1,1} = r \mathbf{v}_{1,1}$, where $\mathbf{v}_{1,1} = [1, 1, \dots, 1]^T$. By similar argument the associated left eigenvector of \mathbf{A}_1 is $\mathbf{w}_{1,1} = [1, 1, \dots, 1]^T$. Similar statements hold for \mathbf{A}_2 .

Define $\tilde{\mathbf{w}}_{1,1}$ to be $\mathbf{w}_{1,1}$ appended with a zeros, and $\tilde{\mathbf{w}}_{2,1}$ to be $\mathbf{w}_{2,1}$ prefixed with a zeros. Now, let us complete the derivations in Section IV using $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$, with $\mathbf{w}_1 \triangleq \tilde{\mathbf{w}}_{1,1} + \tilde{\mathbf{w}}_{2,1}$. Since $\tilde{\mathbf{w}}_{1,1}$ and $\tilde{\mathbf{w}}_{2,1}$ are associated with the same eigenvalue, their sum $\mathbf{w}_1 = [1, 1, \dots, 1]^T$ is also a left eigenvector of \mathbf{A} . Note that \mathbf{w}_1 is not the only positive left eigenvector of \mathbf{A} , as any linear combination $c_1 \tilde{\mathbf{w}}_{1,1} + c_2 \tilde{\mathbf{w}}_{2,1}$ with $c_i > 0$ would also be a positive left eigenvector. For the cycle graphs under consideration, (32) becomes

$$\begin{aligned} \beta'_l &= (\tilde{\mathbf{w}}_{1,1}^T + \tilde{\mathbf{w}}_{2,1}^T) \mathbf{B} \lambda \prod_{j=1}^l g_j \\ &+ \sum_{i=1}^l (\tilde{\mathbf{w}}_{1,1}^T + \tilde{\mathbf{w}}_{2,1}^T) \left(\mathbf{B} \lambda + \mathbf{B}_{ex} \lambda_i^{(ex)} \right) \prod_{j=i+1}^l g_j. \end{aligned} \quad (73)$$

This is equivalent to the other expressions derived in Section IV since $\mathbf{w}_1 = \tilde{\mathbf{w}}_{1,1} + \tilde{\mathbf{w}}_{2,1}$, noting that $r = 1$ here. With these considerations we may use the results of Section IV and later for the allowed reducible \mathbf{A} matrices. Relative to the variable-regular codes addressed in this paper, the cycle graph C_a corresponds to the (a, b) trapping set with $a \geq 2$ and $b = (d_v - 2)a$.

APPENDIX B ADDING LEAVES AND BRANCHES

We now expand the set of multigraphs addressed in this work to include those with leaves and branches, which were previously eliminated by Assumption 7. Thus, the variable nodes in the associated Tanner subgraphs will be permitted to have $d_v - 1$ adjacent degree-one check nodes. In [1], Sun referred to the graphs addressed in this section as “trapping sets with external data nodes.”

We will describe the elementary Tanner graphs of this section from the perspective of their associated multigraphs using the mapping of Lemma 3. Let a *base* graph be a multigraph that meets Assumptions 2, 6, and 7. Such a graph contains one or more cycles. A *leaf* is a vertex of degree one. A *branch* is a vertex of degree two or more outside of the base graph, and is not contained in any cycles.

Example 8. Fig. 5a contains one leaf on a base graph and Fig. 5b contains just three leaves and no base graph. The (18, 8) failing structure of Fig. 9c has leaves A, B, and D and branch C on a base graph that is the (14, 4) trapping set of Fig. 9b.

Assumption 11 (Replacement for Assumption 7). Multigraphs of interest must contain a base graph and zero or more leaves and branches.

We require a base graph because without a base graph the eigenvalues would all be zero, as will become apparent shortly. Our new assumption allows the variable nodes within the associated Tanner subgraphs to neighbor up to $d_v - 1$ degree-one check nodes.

Theorem 14. Let $G = (V, F)$ be a multigraph meeting Assumptions 2, 6, and 11 containing base graph G_B , n leaves, with $n \geq 1$, and zero or more branches. Let $D = (Z, A)$ and D_B be the associated digraphs, created as described in Section III-C. Then, the adjacency matrix $\mathbf{A}(D)$ is a reducible and its spectrum contains the eigenvalues of $\mathbf{A}(D_B)$ and zeros.

Proof: Each additional leaf in G adds one edge to G . Let the edge f_i join leaf v_k to vertex v_j , which must not be a leaf. The edge f_i maps to vertices $z_i, z_i' \in Z$ in D . These vertices are not strongly connected to the digraph as $d_i^+ = 0$ and $d_{i'}^- = 0$, and hence $\mathbf{A}(D)$ must be reducible by Lemma 2.

Every leaf in G creates an all-zero column in $\mathbf{A}(D)$ due to $d_{i'}^- = 0$ and an all-zero row due to $d_i^+ = 0$. The all-zero columns may be symmetrically permuted to the front and the all-zero rows to the back to form

$$\mathbf{P} \mathbf{A}(D) \mathbf{P}^T = \begin{pmatrix} \mathbf{0} & \mathbf{Y}_1 & \mathbf{Y}_2 \\ \mathbf{0} & \mathbf{B} & \mathbf{Y}_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad (74)$$

where the block diagonal contains square submatrices. The $n \times n$ zero matrices at both ends of the block diagonal correspond to the n leaves in D .

The eigenvalues of $\mathbf{A}(D)$ are the roots of the characteristic equation $\det(\mathbf{A}(D) - \mu \mathbf{I}) = 0$, which simplifies to $\det(\mathbf{B} - \mu \mathbf{I}) \mu^{2n} = 0$ by use of the expansion by minors along every zero row and column. Thus, the eigenvalues of $\mathbf{A}(D)$ are the eigenvalues of \mathbf{B} and $2n$ zeros. In case G contains both leaves and branches, the removal of one layer of leaves exposes a new layer of leaves and the operations in this paragraph must be repeated until we work down to $\mathbf{B} = \mathbf{A}(D_B)$, which will be irreducible except for the case addressed in Appendix A in which G_B is a cycle graph. ■

By showing that the nonzero eigenvalues are preserved by the addition of leaves and branches, it is simple to argue that Theorem 5 and Corollary 6, which bound the spectral radius of $\mathbf{A}(D)$, still hold if Assumption 7 is replaced by 11.

The only weakness with respect to our prior development is that we can no longer assume that left eigenvector \mathbf{w}_1 is positive as $\mathbf{A}(D)$ may now be reducible. In practice, we have found that the new entries added to \mathbf{w}_1 by the addition of leaves and branches are half zero and half positive. This may be proved by applying the Subinvariance Theorem in [30] to (74). The presence of zero entries in \mathbf{w}_1 weakens our prior claims on the error indicator $\beta_l \triangleq \mathbf{w}_1^T \mathbf{x}_l$. Now, the error indicator is most effective on the variable nodes corresponding to the base graph and less effective on the variable nodes corresponding to the branches and leaves.

Finally, we don't see much practical motivation to predict the error floors of graphs with branches and leaves. We would prefer to run predictions on the base graphs contained within. Since the graphs with branches and leaves have the same spectral radius as their base graph they are no more likely to fail in theory. In fact, they should be less likely to fail as more channel values are involved and more unsatisfied check nodes are working to correct the graph.

TABLE IV

CONNECTED, ELEMENTARY ABSORBING SETS FROM THE SET OF
4-CYCLE-FREE $d_v = 3$ VARIABLE-REGULAR CODES, WITH $r_{max} > 1.3$

(a, b)	Num.	h_{max}	r_{min}	r_{max}
4,0	1	1	2	2
4,2	1	1	1.521	1.521
5,1	1	1	1.829	1.829
5,3	2	4	1.414	1.424
6,0	2	2	2	2
6,2	4	2	1.696	1.729
6,4	4	2	1.348	1.361
7,1	4	1	1.883	1.888
7,3	10	2	1.599	1.665
7,5	6	2	1.298	1.316
8,0	5	2	2	2
8,2	19	2	1.780	1.870
8,4	25	2	1.521	1.622
9,1	19	1	1.911	1.929
9,3	63	2	1.696	1.851
9,5	52	2	1.463	1.592
10,0	19	2	2	2
10,2	113	2	1.829	1.911
10,4	198	2	1.629	1.841
10,6	109	4	1.414	1.570
11,1	114	1	1.929	1.947
11,3	482	2	1.758	1.899
11,5	536	2	1.571	1.836
11,7	197	2	1.379	1.555
12,0	85	2	2	2
12,2	835	2	1.861	1.940
12,4	1,892	2	1.696	1.894
12,6	1,373	2	1.521	1.833
12,8	351	2	1.348	1.545
13,1	839	1	1.941	1.961
13,3	4,541	2	1.799	1.934
13,5	6,374	2	1.645	1.891
13,7	3,159	2	1.481	1.831
13,9	581	2	1.321	1.537
14,0	509	2	2	2
14,2	7,589	2	1.883	1.954
14,4	21,434	2	1.745	1.931
14,6	19,587	2	1.599	1.889
14,8	6,879	2	1.446	1.830
14,10	931	4	1.298	1.532

APPENDIX C

TABLES OF ABSORBING SET STRUCTURE

As opposed to studying the trapping sets that dominate several specific codes, in this section we take an overview of all subgraphs that may become troublesome trapping sets for a class of codes and that meet our conditions. This can serve to put some perspective on the structural parameters that have been discussed in this paper such as: a , b , the index of imprimitivity h , and the spectral radius r .

The variable-regular codes we examine are assumed to be described by Tanner graphs that are 4-cycle-free. We divide the information into tables by their variable-degree d_v . The further conditions we put on the subgraphs of interest are described in Assumptions 2 and 6 and Definition 3; the subgraphs must be elementary, connected and absorbing, respectively.

Rather than find every single graph that meets our parameters we can simplify the search considerably. As we are just interested in a graph's structure as opposed to its labeling we need only identify one of the graphs among several that are isomorphic to each other. This is known as partitioning the graphs into *equivalence classes*.

Graph theory tools such as “geng” [45] can generate non-isomorphic simple graph descriptions very quickly satisfying sets of parameters such as ours. We run this tool once for each row of the tables. Each time we configure it to find undirected graphs of order a , size $(ad_v - b)/2$, and with vertex degrees

TABLE V

CONNECTED, ELEMENTARY ABSORBING SETS FROM THE SET OF
4-CYCLE-FREE $d_v = 4$ VARIABLE-REGULAR CODES

(a, b)	Num.	h_{max}	r_{min}	r_{max}
4,4	1	1	2	2
5,0	1	1	3	3
5,2	1	1	2.629	2.629
5,4	1	1	2.219	2.219
6,0	1	1	3	3
6,2	2	1	2.697	2.710
6,4	3	1	2.355	2.367
6,6	2	2	2	2
7,0	2	1	3	3
7,2	7	1	2.744	2.762
7,4	11	2	2.449	2.480
7,6	4	1	2.159	2.160
8,0	6	2	3	3
8,2	28	2	2.778	2.805
8,4	50	2	2.525	2.585
8,6	28	2	2.272	2.296
8,8	5	2	2	2
9,0	16	1	3	3
9,2	126	1	2.805	2.850
9,4	285	2	2.584	2.728
9,6	177	2	2.355	2.429
9,8	27	1	2.126	2.141
10,0	59	2	3	3
10,2	719	2	2.826	2.886
10,4	1,915	2	2.629	2.821
10,6	1,404	2	2.422	2.648
10,8	298	2	2.219	2.313
10,10	19	2	2	2
11,0	265	1	3	3
11,2	4,721	1	2.843	2.903
11,4	14,569	2	2.667	2.852
11,6	12,458	2	2.478	2.788
11,8	3,231	2	2.295	2.458
11,10	208	1	2.104	2.127

TABLE VI

CONNECTED, ELEMENTARY ABSORBING SETS FROM THE SET OF
4-CYCLE-FREE $d_v = 5$ VARIABLE-REGULAR CODES, WITH $r_{max} > 2.6$

(a, b)	Num.	h_{max}	r_{min}	r_{max}
5,5	1	1	3	3
5,7	1	1	2.629	2.629
6,0	1	1	4	4
6,2	1	1	3.693	3.693
6,4	2	1	3.360	3.403
6,6	4	1	3	3.112
6,8	5	1	2.697	2.767
7,1	1	1	3.875	3.875
7,3	5	1	3.596	3.669
7,5	14	1	3.307	3.427
7,7	23	1	3	3.130
7,9	25	1	2.744	2.827
8,0	3	1	4	4
8,2	16	1	3.775	3.827
8,4	68	1	3.522	3.673
8,6	165	1	3.271	3.465
8,8	252	2	3	3.208
8,10	232	2	2.778	2.918
8,12	124	2	2.525	2.619
9,1	28	1	3.904	3.905
9,3	276	1	3.693	3.769
9,5	1,151	2	3.464	3.665
9,7	2,541	2	3.243	3.521
9,9	3,284	2	3	3.289
9,11	2,541	2	2.805	3.071
9,13	1,150	2	2.584	2.751
10,0	60	2	4	4
10,2	1,188	2	3.822	3.870
10,4	8,435	2	3.626	3.789
10,6	27,706	2	3.421	3.810
10,8	49,991	2	3.220	3.717
10,10	53,884	2	3	3.440
10,12	35,721	2	2.826	3.191
10,14	14,308	2	2.629	3.009
10,16	3,224	2	2.422	2.652

in the interval $[[d_v/2], d_v]$. The size we have specified is a direct result of Euler's handshaking lemma. By limiting the tool output to simple graphs, we eliminate multigraphs of girth

2 and their equivalent Tanner subgraphs of girth 4. The range of vertex degrees specified ensures that we get absorbing sets. We then process each graph in a custom tool that converts it to the adjacency matrix of the associated directed graph to find its spectral radius r and index of imprimitivity h .

Table IV shows the graphs found based on a code with $d_v = 3$. The graphical equivalence classes found are divided into rows by their (a, b) parameters listed in the first column. The second column of each row presents the number of equivalence classes found that meet the (a, b) parameters and all our other assumptions. The third column shows the maximum index of imprimitivity h over the (a, b) equivalence classes. The fourth and fifth columns show the minimum and maximum spectral radius r over the (a, b) equivalence classes. To save space we have left out the weakest (a, b) pairs with $r_{max} \leq 1.3$. Tables V and VI present similar results based on codes with $d_v = 4$ and 5, respectively, omitting the (a, b) pairs with $r_{max} \leq 2.6$ for $d_v = 5$.

By enumerating absorbing sets, the focus of this section is the trapping sets of the saturating SPA decoder. These tables present enough information to comment on Schlegel and Zhang's approximation to the spectral radius r of absorbing sets, which is $r \approx d_v - 1 - b/a$ [20]. We find empirically that this approximation does not overestimate the spectral radius, but generally underestimates it. For most (over 90%) of the $d_v = 3$ equivalence classes, this approximation ranges from 0% to 6% below the correct spectral radius value. The extreme relative error we found was -20.2% for a $(13, 7)$ absorbing set in Table IV, -12.0% for an $(11, 6)$ absorbing set in Table V, and -13.9% for a $(10, 8)$ absorbing set in Table VI.

ACKNOWLEDGMENT

The authors would like to thank Aravind Iyengar, Christian Schlegel, Roxana Smarandache, and Xiaojie Zhang for their helpful discussions and encouragement. Further, the authors are indebted to Xiaojie Zhang for verifying several of our low floor results in the AWGN channel with an independent implementation of the non-saturating SPA decoder. Also, we thank Yang Han and William Ryan for providing the QC parity check matrix.

REFERENCES

- [1] J. Sun, "Studies on graph-based coding systems," Ph.D. dissertation, Ohio State Univ., Columbus, 2004.
- [2] R. G. Gallager, "Low-density parity-check codes," *IRE Trans. on Inform. Theory*, pp. 21–28, Jan. 1962.
- [3] —, *Low-Density Parity-Check Codes*. Cambridge, MA: M.I.T. Press, 1963.
- [4] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, Aug. 1996.
- [5] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping Univ., Sweden, Apr. 1996.
- [6] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite length analysis of low-density parity-check codes," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.
- [7] D. J. C. MacKay and M. J. Postol, "Weaknesses of Margulis and Ramanujan–Margulis low-density parity-check codes," in *Proc. of the 2nd Irish Conf. on the Math. Foundations of Comput. Sci. and Inf. Tech. (MFCSIT)*, ser. Electronic Notes in Theoretical Comput. Sci., vol. 74, Galway, 2003.
- [8] T. Richardson, "Error-floors of LDPC codes," in *Proc. of the 41st Annu. Allerton Conf.*, Oct. 2003, pp. 1426–1435.
- [9] C.-C. Wang, S. Kulkarni, and H. Poor, "Finding all small error-prone substructures in LDPC codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 5, pp. 1976–1999, May 2009.
- [10] S. Abu-Surra, D. DeClercq, D. Divsalar, and W. E. Ryan, "Trapping set enumerators for specific LDPC codes," in *Inf. Theory and Appl. Workshop (ITA)*, San Diego, Feb. 2010.
- [11] M. Karimi and A. H. Banihashemi, "An efficient algorithm for finding dominant trapping sets of LDPC codes," Aug. 2011. [Online]. Available: <http://arxiv.org/abs/1108.4478v1>
- [12] O. Milenkovic, E. Soljanin, and P. Whiting, "Asymptotic spectra of trapping sets in regular and irregular LDPC code ensembles," *IEEE Trans. Inf. Theory*, vol. 53, no. 1, pp. 39–55, Jan. 2007.
- [13] L. Dolecek, Z. Zhang, V. Anantharam, M. J. Wainwright, and B. Nikolić, "Analysis of absorbing sets for array-based LDPC codes," in *Proc. IEEE Int. Conf. on Commun.*, Glasgow, Jun. 2007, pp. 6261–6268.
- [14] Y. Han and W. E. Ryan, "Low-floor decoders for LDPC codes," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1663–1673, Jun. 2009.
- [15] S. Ländner and O. Milenkovic, "Algorithmic and combinatorial analysis of trapping sets in structured LDPC codes," in *Proc. Int. Conf. on Wireless Netw., Commun. and Mobile Computing*, Maui, Jun. 2005, pp. 630–635 vol.1.
- [16] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. J. Wainwright, "Lowering LDPC error floors by postprocessing," in *Proc. IEEE Global Telecommun. Conf.*, New Orleans, Dec. 2008, pp. 1–6.
- [17] A. Vila Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proc. IEEE Int. Conf. on Commun.*, Glasgow, Jun. 2007, pp. 932–937.
- [18] E. Sharon, O. Fainzilber, and S. Litsyn, "Decreasing error floor in LDPC codes by parity-check matrix extensions," in *Proc. IEEE Int. Symp. on Inform. Theory*, Seoul, Jul. 2009, pp. 374–378.
- [19] J. Sun, O. Y. Takeshita, and M. P. Fitz, "Analysis of trapping sets for LDPC codes using a linear system model," in *Proc. of the 42nd Annu. Allerton Conf.*, Oct. 2004, pp. 1701–1702.
- [20] C. Schlegel and S. Zhang, "On the dynamics of the error floor behavior in (regular) LDPC codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3248–3264, Jul. 2010.
- [21] J. Brevik and M. E. O'Sullivan, "The sum-product algorithm for degree-2 check nodes and trapping sets," Oct. 2010, unpublished.
- [22] J. Thorpe, "Low-complexity approximations to belief propagation for LDPC codes," Oct. 2002, unpublished, online.
- [23] J. Chen and M. P. Fossorier, "Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, Taipei, Nov. 2002, pp. 1378–1382.
- [24] J. Zhao, F. Zarkeshvari, and A. H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density parity-check (LDPC) codes," *IEEE Commun. Lett.*, vol. 53, no. 4, pp. 549–554, Apr. 2005.
- [25] Z. Zhang, L. Dolecek, M. Wainwright, V. Anantharam, and B. Nikolić, "Quantization effects in low-density parity-check decoders," in *Proc. IEEE Int. Conf. on Commun.*, Glasgow, Jun. 2007, pp. 6231–6237.
- [26] Z. Zhang, L. Dolecek, B. Nikolić, V. Anantharam, and M. Wainwright, "Investigation of error floors of structured low-density parity-check codes by hardware emulation," in *Proc. IEEE Global Telecommun. Conf.*, San Francisco, Nov. 2006, pp. 1–6.
- [27] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge Univ. Press, 1985.
- [28] C. Meyer, *Matrix Analysis and Applied Linear Algebra*. Philadelphia: SIAM, 2000.
- [29] H. Minc, *Nonnegative Matrices*. New York: John Wiley & Sons, 1988.
- [30] E. Seneta, *Non-negative Matrices and Markov Chains*. New York: Springer, 1981.
- [31] F. Harary and R. Z. Norman, "Some properties of line digraphs," *Rend. del Circ. Mat. di Palermo*, vol. 9, no. 2, pp. 161–168, 1960.
- [32] J. L. Gross and J. Yellen, *Graph Theory and Its Applications*. Boca Raton, Florida: Chapman & Hall/CRC, 2006.
- [33] R. J. Trudeau, *Introduction to Graph Theory*. Kent, Ohio: Kent State Univ. Press, 1976.
- [34] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, San Antonio, Nov. 2001, pp. 1036–1036E.
- [35] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution," in *Proc. IEEE Global Telecommun. Conf.*, vol. 2, San Antonio, Nov. 2001, pp. 1021–1025.

- [36] G. Richter, G. Schmidt, M. Bossert, and E. Costa, "Optimization of a reduced-complexity decoding algorithm for LDPC codes by density evolution," in *Proc. IEEE Int. Conf. on Commun.*, vol. 1, Seoul, May 2005, pp. 642–646.
- [37] S. Laendner, T. Hehn, O. Milenkovic, and J. B. Huber, "The trapping redundancy of linear block codes," *IEEE Trans. Inf. Theory*, vol. 55, no. 1, pp. 53–63, Jan. 2009.
- [38] C. A. Cole, S. G. Wilson, E. K. Hall, and T. R. Giallorenzi, "A general method for finding low error rates of LDPC codes," 2006. [Online]. Available: <http://arxiv.org/abs/cs/0605051>
- [39] Y. Zhang and W. E. Ryan, "Toward low LDPC-code floors: A case study," *IEEE Trans. Commun.*, vol. 57, no. 6, pp. 1566–1573, Jun. 2009.
- [40] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.
- [41] S.-Y. Chung, G. D. Forney, Jr., T. J. Richardson, and R. L. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.
- [42] T. Kailath, *Linear Systems*. Upper Saddle River, NJ: Prentice Hall, 1980.
- [43] M. Fu, "On Gaussian approximation for density evolution of low-density parity-check codes," in *Proc. IEEE Int. Conf. on Commun.*, vol. 3, Istanbul, Jun. 2006, pp. 1107–1112.
- [44] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge, UK: Cambridge Univ. Press, 2009.
- [45] B. D. McKay. (2009) nauty user's guide (version 2.4). [Online]. Available: <http://cs.anu.edu.au/~bdm/nauty/>